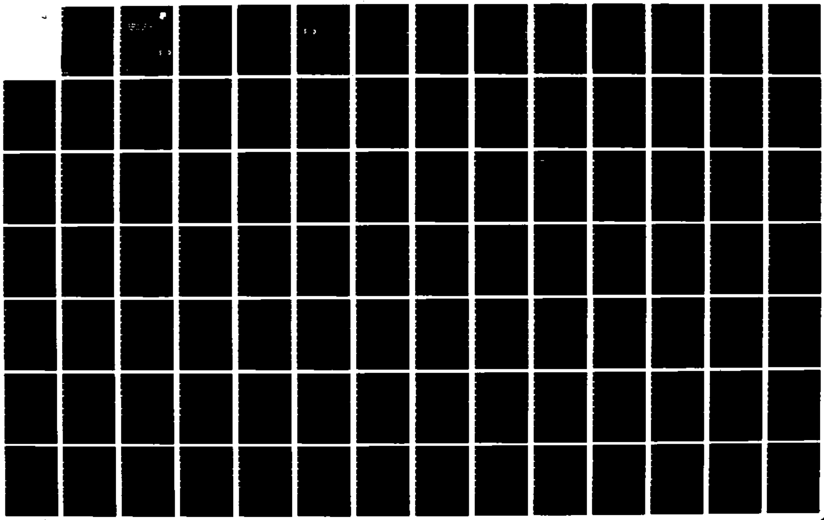
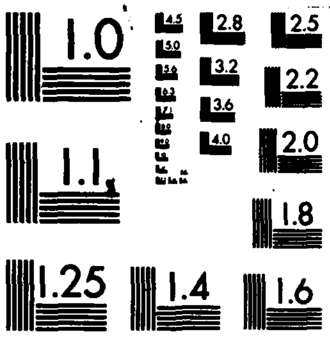



1.0
4.5
5.0
5.6
6.3
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

(12)

AD A137956

RADC-TR-83-175, Vol II (of three)
Final Technical Report
July 1983

SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS Guidebook for Software Quality Measurement

Boeing Aerospace Company

Thomas P. Bowen, Jonathan V. Post, Juitien Tsai, P. Edward Presson and Robert L. Schmidt

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

RADC-TR-83-175, Vol II (of three) has been reviewed and is approved for publication.

APPROVED: Joseph P. Cavano

JOSEPH P. CAVANO
Project Engineer

APPROVED: [illegible]

RONALD S. RAPOSO
Acting Chief, Command & Control Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-83-175, Vol II (of three)	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS Guidebook for Software Quality Measurement		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report October 80 - March 83
		6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) Thomas P. Bowen Jonathan V. Post Juitien Tsai P. Edward Presson Robert L. Schmidt		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0330
9. PERFORMING ORGANIZATION NAME AND ADDRESS Boeing Aerospace Company PO Box 3999 Seattle WA 93124		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 55812030
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (COEE) Griffiss AFB NY 13441		12. REPORT DATE July 1983
		13. NUMBER OF PAGES 280
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Joseph P. Cavano (COEE)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software Quality	Software Survivability
Software Metrics	Software Expandability
Software Measurement	
Software Interoperability	
Software Reusability	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Software metrics (or measurements) which are used to indicate and predict levels of software quality were extended from previous research to include considerations for distributed computing systems. Aspects of the products of software life-cycle activities which could affect the quality levels of software, and metrics to measure them, were identified. Two new quality factors, survivability and expandability, were validated. A Guidebook for Software Quality Measurement was produced to aid in setting quality goals,

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

applying metric measurements, and making quality level assessments. New metrics for interoperability and reusability were also included in the guidebook. This volume describes the application of quality metrics to distributed systems and provides guidance for AF acquisition managers. The guidebook provides guidance for specifying and measuring the desired level of quality in a software product.

Accession For	
NTIS GRA&I	✓
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

PREFACE

This document is Volume II of three of the Final Technical Report (CDRL A003) for the Quality Metrics for Distributed Systems contract, Number F30602-80-C-0330. The contract was performed for Rome Air Development Center (RADC) to provide methodology and technical guidance on software quality metrics to Air Force software acquisition managers.

The final report consists of three volumes as follows:

Volume I - Software Quality Measurement for Distributed Systems - Final Report

Volume II - Guidebook for Software Quality Measurement

Volume III - Distributed Computing Systems: Impact on Software Quality

The objective of this contract was to develop techniques to measure and predict software quality with a perspective on distributed systems. The techniques developed were to be assembled into a "Handbook" which describes the step-by-step procedures required to implement the quality measurements. Various methods of assembling a handbook were investigated and it was decided that the best approach would be to use the "Software Quality Measurement Manual" (RADC-TR-80-109), produced by a prior quality metric research contract, as the baseline. Volume II of this final report is therefore an update of RADC-TR-80-109, incorporating results of this contract and the results of contract F30602-80-C-0265, "Software Interoperability and Reusability". In addition, many editorial changes and corrections were made, and all metric worksheets, tables, and definitions are included as appendices so that all material required to implement software quality measurements is included in this document.

Volume I of this report describes the results of the research effort conducted under this contract. Volume III of this report summarizes the analysis performed to determine the effect of distributed computing systems on the quality metrics technology.

TABLE OF CONTENTS

Section		Page
1.0	INTRODUCTION	1-1
	1.1 Purpose	1-1
	1.2 Scope	1-2
	1.3 Quality Measurement in Perspective	1-2
	1.4 Guidebook Organization	1-4
	1.5 Recommended Use of Guidebook	1-7
2.0	IDENTIFYING SOFTWARE QUALITY REQUIREMENTS	2-1
	2.1 Introduction	2-1
	2.2 Identifying Important Quality Factors	2-4
	2.2.1 Procedures	2-4
	2.2.2 An Example of Factors Specification	2-13
	2.3 Identifying Critical Software Attributes	2-16
	2.3.1 Procedures	2-16
	2.3.2 Example of Identifying Software Criteria	2-23
	2.4 Establishing Quantifiable Goals	2-25
	2.4.1 Procedures	2-25
	2.4.2 Example of Metrics	2-36
	2.5 Evaluation of Development Plan	2-38
3.0	APPLYING METRICS	3-1
	3.1 When to Take Measurements	3-1
	3.2 Sources of Quality Information	3-3
	3.3 Application of the Metrics	3-5
	3.4 Techniques for Applying Metrics	3-9
4.0	ASSESSING THE QUALITY OF THE SOFTWARE PRODUCT	4-1
	4.1 Introduction	4-1
	4.2 Inspector's Assessment	4-1
	4.3 Sensitivity Analysis	4-2
	4.4 Use of Normalization Function to Assess Quality	4-4
	4.5 Reporting Assessment Results	4-13

TABLE OF CONTENTS

Section		Page
REFERENCES		4-15
APPENDICES		
A	Metric Worksheets	A-1
B	Metric Tables	B-1
C	Metric Explanations	C-1

LIST OF FIGURES

Figure Number		Page
2.1-1	Software Quality Framework	2-2
2.2-1	Benefit Tradeoff: Quality Costs vs Cost Savings	2-10
3.1-1	Timing of Metrics Application	3-2
3.2-1	Sources of Quality Metric Data	3-4
3.3-1	Application of the Metric Worksheets	3-8
4.4-1	Normalization Function for Flexibility During Design	4-11
4.4-2	Determination of Level of Confidence	4-12

LIST OF TABLES

Table Number		Page
1.3-1	How Software Metrics Complement Quality Assurance	1-6
1.5-1	Index of Three Approaches to Specifying and Assessing Software Quality	1-8
2.1-1	Software Quality Factors and Criteria	2-3
2.2-1	Software Quality Requirements Form	2-5
2.2-2	Examples of System Characteristics and Related Quality Factors	2-7
2.2-3	Relationship of Quality Factors to Life-Cycle Phases	2-9
2.2-4	Relationships Between Software Quality Factors	2-11
2.2-5	Typical Factor Tradeoffs	2-12
2.3-1	Software Criteria and Related Quality Factors	2-17
2.3-2	Definitions of Software Criteria	2-20
2.3-3	Software Criteria to Factor Relationships	2-24
2.4-1	Quality Factor Ratings	2-27
2.4-2	Quality Factor Rating Explanation	2-29
2.4-3	Quality Metrics Related to Factors	2-30
2.4-4	Software Metric to Factor Relationship—Subset	2-37
3.3-1	Metric Worksheet Correlation	3-6
4.4-1	Normalization Functions	4-7

SECTION 1
INTRODUCTION

1.1 PURPOSE

There has been an increased awareness in recent years of the critical problems that have been encountered in the development of large scale software systems. These problems include not only the cost and schedule overruns typical of development efforts and the poor performance of the systems once they are delivered, but also the high cost of maintaining the systems, the lack of portability, and the high sensitivity to changes in requirements.

The government and DoD in particular, as customers of many large scale software system developments, have sponsored many research efforts aimed at attacking these problems. For example, the efforts related to the development of a standard DoD programming language, software development techniques, and development tools and aids all provide partial solutions to the above problems by encouraging a more disciplined approach to the development of software and therefore a more controlled development process.

A related research thrust which has been recently funded by DoD is the area of software quality metrics. The research in this area has resulted in the development and validation of a number of metrics which quantitatively measure various attributes of software which are related to different aspects of software quality.

The potential of the software metric concepts can be realized by use in software procurement. Their use enables an acquisition manager to quantitatively specify the desired level of quality for the software product and to periodically measure the achieved level of quality throughout the software development process. Their effect on a quality assurance program is to provide a more disciplined, engineering approach to quality assurance and to provide a mechanism for taking a life cycle viewpoint of software quality. The benefits derived from their application are realized in life cycle cost reduction and improved software quality resulting from added visibility for management control.

The purpose of this guidebook is to present a complete set of procedures and guidelines for introducing and utilizing current software quality metric techniques for a software procurement associated with large scale software system developments. These procedures and guidelines encompass:

1. How to identify and specify software quality requirements;
2. How and when to apply software metrics; and
3. How to interpret the information obtained from the application of the metrics.

1.2 SCOPE

This guidebook incorporates the results of research conducted in support of Rome Air Development Center (RADC) in the area of quality metrics for distributed systems and software interoperability and reusability. It is an update of the "Software Quality Measurement Manual" previously produced under contract number F30602-78-C-0216 and published as RADC-TR-80-109, Volume II (of two). Software quality metric information for the quality factors of survivability, expandability, interoperability and reusability has been added; information for use with distributed systems has been added; editorial changes have been made; the metric worksheets have been refined, reorganized, and placed in an appendix; and metric tables and definitions have been added to the guidebook (appendices) for ease of use.

While some aspects of the technology of software quality metrics require further research, those portions which can currently provide benefit to a software acquisition manager are emphasized in this guidebook. Guidelines and procedures for using the software metrics are described. The guidelines and procedures are presented in such a way as to facilitate their application when using this guidebook for a software development project. All of the procedures are described as manual processes, however, where automated software tools could be used to compliment or enhance the process, the tools are identified.

Throughout this document the terms guidebook, handbook and manual are used interchangeably.

1.3 QUALITY MEASUREMENT IN PERSPECTIVE

The evolution during the past decade of modern programming practices, structured, disciplined development techniques and methodologies, and requirements for more structured, effective documentation has increased the feasibility of effective measurement of software quality. However, before the potential of measurement techniques could be realized a framework or model of software quality had to be constructed. An established model, which at one level provides a user or management oriented view of quality, is described in Section 2 of this guidebook in the perspective of how it can be used to establish software quality requirements for a specific application.

The actual measurement of software quality, described in Section 3.0, is accomplished by applying software metrics (or measurements) to the documentation and source code produced during software development. These measurements are part of the established model of software quality, and through that model they can be related to various user-oriented aspects of software quality.

The metrics can be classified according to three categories: anomaly-detecting, predictive, and acceptance.

- Anomaly-detecting metrics identify deficiencies in documentation or source code. These deficiencies usually are corrected to improve the quality of the software product. Standards enforcement is a form of anomaly-detecting metrics.

- Predictive metrics are measurements of the soundness of the design and implementation. These measurements are concerned with form, structure, density, and complexity type attributes. They provide an indication of the quality that will be achieved in the end product-based on the nature of the application and the design and implementation strategies.

- Acceptance metrics are measurements that are applied to the end product to assess the final compliance with requirements. Tests are a form of acceptance-type measurements.

The metrics described and used in this guidebook are either anomaly-detecting or predictive. They are applied during the software development phases to assist in identification of quality problems early in the life cycle so that corrective actions can be taken early when they are more effective and economical and to enable a prediction of the quality level expected for the final product.

The measurement concepts complement current quality assurance practices; they are not a replacement for current techniques utilized in normal quality assurance programs. For example, a major objective of quality assurance is to assure compliance with user/customer requirements. The software quality metric concepts described in this guidebook provide a methodology for the user/customer to specify life-cycle-oriented quality requirements, usually not considered, and to provide a mechanism for measuring whether or not those requirements have been attained. A function usually performed by quality assurance personnel is a review/audit of software products produced during software development. The software metrics add formality and quantification to these document and code reviews. The metric concepts also provide a vehicle for early involvement in the development process since there are metrics which apply to the requirements and design documents produced early in the development.

Testing is usually oriented toward evaluating performance (reliability, usability, performance, efficiency, etc.). The metrics can assist in evaluating other qualities such as maintainability, portability, flexibility, etc.

A summary of how the software metric concepts complement quality assurance activities is provided in Table 1.3-1. This is based on the quality assurance program requirements identified in MIL-S-52779. These concepts will be further explained and illustrated in the subsequent sections of this guidebook.

1.4 GUIDEBOOK ORGANIZATION

The guidebook has been organized as a handbook for use in software acquisition. Section 1 provides introductory information and how the guidebook is to be used.

Section 2 defines the software quality model and describes a methodology for using this model to establish software quality requirements or goals for a software development project.

Section 3 describes procedures for measuring the quality of the software. These procedures cover what to measure, when to measure, and how to measure.

Section 4 describes procedures for utilizing the information provided by the measurements to make assessments of the quality of the software and recommends what information to present to various personnel involved in the development.

Appendix A contains the metric worksheets used for collecting data.

Appendix B contains the metric tables used for calculating metric scores during the various measurement periods.

Appendix C contains a detailed description of the metric elements.

Table 1.3-1 How Software Metrics Complement Quality Assurance

MIL-S-52779 QUALITY ASSURANCE PROGRAM REQUIREMENTS	IMPACT OF SOFTWARE QUALITY METRIC CONCEPTS
Assure Compliance with Requirements	Adds software quality requirements
Identify Software Deficiencies	Anomaly-detecting metrics
Provide Configuration Management	No impact
Conduct Test	Assists in evaluation of other qualities
Provide Library Controls	No impact
Review Computer Program Design	Predictive metrics
Assure Software Documentation Requirement Compliance	Metrics assist in evaluation of documentation as well as code
Conduct Reviews and Audits	Procedures for applying metrics (in form of worksheets) formalizes inspection process
Provide Tools/Techniques/Methodology for Quality Assurance	This manual describes methodology of using metrics
Provide Subcontractor Control	Same as above for all requirements

1.5 RECOMMENDED USE OF GUIDEBOOK

The software quality metric concepts can be applied at several levels. In an acquisition manager/contractor environment, there are three approaches for using the metric concepts. They are:

1. The acquisition manager's staff can establish software quality requirements or goals and apply metrics to the delivered software products.
2. The development manager's staff can apply metrics to software products during development and report them to the acquisition manager during reviews.
3. An independent Quality Assurance or Independent Verification and Validation (IV&V) contractor can apply metrics to delivered software products and report them to the acquisition manager.

Within the software development project organization, there are two approaches for using the metric concepts. They are:

1. The quality assurance personnel can apply the metrics as an independent assessment of the quality of the software being produced and report them to the software development manager.
2. The software development personnel can apply the metrics during walk-throughs and reviews and report them to the software development manager.

This guidebook is oriented toward those personnel who will be applying the quality metrics concepts (either quality assurance or development personnel) and recommends three approaches to both establishing the quality requirements (Section 2) and making a quality level assessment (Section 4). The three approaches (an index is provided in Table 1.5-1) in each area are presented in order of increasing formality of the relationship between quality requirements and metrics, i.e., in order of increasing quantification. The order of presentation also relates to an increasing requirement for experience with the concepts by the personnel applying the concepts. Thus, the approaches can be used as a phased implementation plan of incorporating the metric concepts into the quality assurance functions.

Table 1.5-1 Index of Three Approaches to Specifying and Assessing Software Quality

APPROACH (LEVEL OF FORMALITY)	SPECIFYING SOFTWARE QUALITY	APPLYING MEASUREMENTS	ASSESSING THE QUALITY OF THE PRODUCT
1	Procedures for identifying important quality factors (Paragraph 2.2)	PROCEDURES FOR APPLYING THE METRIC WORKSHEETS (SECTION 3)	Procedures for the inspector's assessment (Paragraph 4.2)
2	Procedures for identifying critical software attributes (Paragraph 2.3)		Procedures for performing sensitivity analysis (Paragraph 4.3)
3	Procedures for establishing quantifiable goals (Paragraph 2.4)		Procedures for use of normalization function (Paragraph 4.4)

This guidebook is recommended to the personnel applying the metric concepts. Additional information and definitions can be found in:

"Factors in Software Quality", 3 volumes, RADC-TR-77-369, Nov 1977. (MCCA77)
"Software Quality Metrics Enhancements", 2 volumes, RADC-TR-80-109, April 1980
"Software Interoperability and Reusability-Final Report".
"Software Quality Measurement for Distributed Systems - Final Report", Volume I of this report.
"Distributed Computing Systems: Impact on Software Quality", Volume III of this report.

These references are recommended to the personnel applying the metrics for familiarization with the underlying concepts. They can also be referred to periodically for definitions and explanations.

SECTION 2
IDENTIFYING SOFTWARE QUALITY REQUIREMENTS

2.1 INTRODUCTION

The primary purpose of using software quality metrics in a software acquisition is to improve the quality of the software product by specifying software quality requirements and by measuring and predicting achieved software quality. The concepts can improve quality since they are based on achieving a positive influence on the product.

This section addresses the task of identifying software quality requirements and establishing quantifiable goals. These requirements are in addition to the functional, performance, cost, and schedule requirements normally specified for software development. The fact that the goals established are related to the quality of the end product should, in itself, provide some positive influence. Past research has shown that goal-directed system development is effective. (WEIN72)

The vehicle for establishing the requirements is the hierarchical model of software quality defined in (CAVA78). This model, shown in Figure 2.1-1, has at its highest level a set of software quality factors which are user/management-oriented terms and represent the characteristics which comprise software quality. At the next level, for each quality factor, there is a set of criteria which are the attributes that, if present in the software, provide the characteristics represented by the quality factors. The criteria, then, are software-related terms. Table 2.1-1 identifies the thirteen quality factors, the thirty quality criteria, and their relationships. At the lowest level of the model are the metrics which are quantitative measures of the software attributes defined by the criteria. In a sense there is a still lower level of the model – the metric elements. Several metric elements, completed at several points in the software life-cycle, may be combined in calculations for a single metric. Appendix B, Metric Tables, identifies the metrics and metric elements.

The procedures for establishing the quality requirements for a particular software system utilize this model and are described as a three level approach; the levels correspond to the hierarchical levels of the software quality model. The first level establishes the quality factors that are important. The second level identifies the critical software attributes.

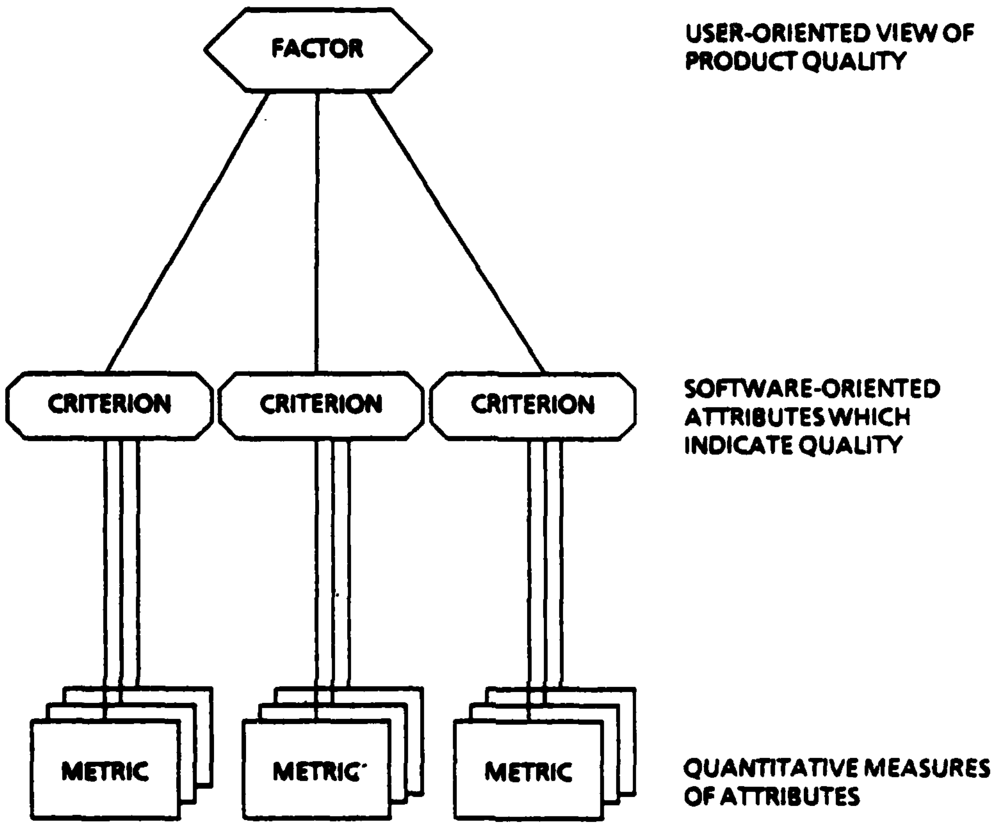

FACTOR
USER-ORIENTED VIEW OF PRODUCT QUALITY
CRITERION
CRITERION
CRITERION
SOFTWARE-ORIENTED ATTRIBUTES WHICH INDICATE QUALITY
METRIC
METRIC
METRIC
QUANTITATIVE MEASURES OF ATTRIBUTES

Figure 2.1-1 Software Quality Framework

Table 2.1-1 Software Quality Factors and Criteria

QUALITY FACTOR (USER ORIENTED) / QUALITY CRITERIA (SOFTWARE OR SOFTWARE PRODUCTION OREINTED)	SOFTWARE OPERATION						SOFTWARE REVISION			SOFTWARE TRANSITION			
	CORRECTNESS	RELIABILITY	EFFICIENCY	INTEGRITY	USABILITY	SURVIVABILITY	MAINTAINABILITY	VERIFIABILITY	FLEXIBILITY	PORTABILITY	REUSABILITY	INTEROPERABILITY	EXPANDABILITY
• ACCURACY		X											
• ANOMALY MANAGEMENT		X				X							
• APPLICATION INDEPENDENCE											X		
• AUGMENTABILITY												X	X
• AUTONOMY						X							
• COMMONALITY												X	
• COMMUNICATIVENESS					X							X	
• COMPLETENESS	X												
• CONCISENESS							X						
• CONSISTENCY	X	X					X						
• DISTRIBUTEDNESS						X							
• DOCUMENT ACCESSIBILITY											X		
• EFFECTIVENESS			X										
• FUNCTIONAL OVERLAP												X	
• FUNCTIONAL SCOPE											X		
• GENERALITY									X		X		X
• INDEPENDENCE										X	X	X	
• MODULARITY						X	X	X	X	X	X	X	X
• OPERABILITY					X								
• RECONFIGURABILITY						X							
• SELF-DESCRIPTIVENESS							X	X	X	X	X		
• SIMPLICITY	X	X					X	X	X		X		X
• SPECIFICITY	X							X					X
• SYSTEM ACCESSIBILITY				X									
• SYSTEM CLARITY											X		
• SYSTEM COMPATIBILITY												X	
• TRACEABILITY	X												
• TRAINING					X								
• VIRTUALITY				X	X								X
• VISIBILITY					X		X	X					

The third level identifies the metrics that will be applied and establishes quantitative ratings for the quality factors.

Once the quality requirements have been determined by following the procedures described in the subsequent paragraphs, they are to be transmitted to the development team. In a formal acquisition manager/contractor environment, the Request for Proposal (RFP) is the medium for identifying these requirements. The results of following the procedures should be incorporated in the RFP. If the development is being done internally, the quality requirements should be documented in the same form as the other system requirements. A briefing emphasizing the intent of the inclusion of the quality requirements can also be conducted.

2.2 IDENTIFYING IMPORTANT QUALITY FACTORS

2.2.1 Procedures

The basic tool utilized in identifying the important quality factors is the Software Quality Requirements Form shown in Table 2.2-1. The formal definitions of each of the thirteen factors are provided on that form.

A briefing, using the tables and figures contained in this paragraph, should be conducted for the decision makers in order to solicit their responses to the survey. The decision makers may include the acquisition manager, the user/customer, the development manager, and the quality assurance manager. To complete the survey the following five procedures are recommended.

1a. <u>Consider Basic Characteristics of the Application</u>
The software quality requirements for each system are unique and are influenced by system or application-dependent characteristics. There are basic characteristics which affect the quality requirements and each software system must be evaluated for its basic characteristics. Table 2.2-2 provides a list of some of these basic characteristics. For example, if the system is being developed in an environment in which there is a high rate of technical breakthroughs in hardware design, portability should take on an added significance. If the expected life cycle of the system is long, maintainability and

Table 2.2-1 Software Quality Requirements Form

The 13 quality factors listed below represent aspects of software quality which are currently thought to be important. Indicate whether you consider each factor to be Very Important (VI), Important (I), Somewhat Important (SI), or Not Important (NI) as design goals in the system you are currently working on or planning.

RESPONSE	FACTORS	DEFINITION
________	CORRECTNESS	Extent to which the software satisfies its specifications and fulfills the user's mission objectives.
________	RELIABILITY	Probability that the software will perform its logical operations in the specified environment without failure.
________	EFFICIENCY	Degree of utilization of resources (processing time, storage, communication time) in performing functions.
________	INTEGRITY	Extent to which unauthorized access to the software or data can be controlled.
________	USABILITY	Effort for training and software operation - familiarization, input preparation, execution, output interpretation.
________	SURVIVABILITY	Probability that the software will continue to perform or support critical functions when a portion of the system is inoperable.
________	MAINTAINABILITY	Average effort to locate and fix a software failure.
________	VERIFIABILITY	Effort to verify the specified software operation and performance.
________	FLEXIBILITY	Effort to extend the software missions, functions, or data to satisfy other requirements.
________	PORTABILITY	Effort to convert the software for use in another operating environment (hardware configuration, software system environment).
________	REUSABILITY	Effort to convert a software component for use in another application.
________	INTEROPERABILITY	Effort to couple the software of one system to the software of another system.
________	EXPANDABILITY	Effort to increase software capability or performance by enhancing current functions or adding new functions/data.

Title:	Name:	Signature:

Table 2.2-2 Example of System Characteristics and Related Quality Factors

SYSTEM CHARACTERISTIC	QUALITY FACTOR
If human lives are affected	Reliability Correctness Verifiability Survivability
Long life cycle	Maintainability Expandability
Experimental system high rate of change	Flexibility
High technology in hardware design	Portability
Many system changes over life cycle	Reusability Expandability
Real time application	Efficiency Reliability Correctness
On-board computer application	Efficiency Reliability Correctness Survivability
Processes classified information	Integrity
Interrelated systems	Interoperability

expandability become cost-critical considerations. If the application is an experimental system where the software specifications will have a high rate of change, flexibility and expandability in the software product are highly desirable. If the functions of the system are expected to be required for a long time, while the system itself may change considerably, reusability and expandability are of prime importance in those modules which implement the major functions of the system. With the advent of more computer networks and communication capabilities, more systems are being required to interface with other systems and the concept of interoperability is extremely important. With distributed computing systems, more attention is given to providing some essential computational services even when some subsystems are inoperable, and the concept of survivability is extremely important. For systems with long life-cycles (e.g., 15-20 years for a major weapon system) some provisions must be made for incorporating new hardware (add-on memory or peripherals) or new software (upgraded operating system), and the concept of expandability becomes crucial. These and other system characteristics should be considered when identifying the important quality factors.

If system level quality requirements have already been established, refer to Section 3.2 of Volume I of this report for aids in allocating the system quality requirements to the software level and in identifying important software quality factors.

1b. <u>Consider Life Cycle Implications</u>

The thirteen quality factors identified on the Software Quality Requirements Form (Table 2.2-1) can be grouped according to three life cycle activities associated with a delivered software product. These three activities are product operation, product revision, and product transition. The relationship of the quality factors to these activities is shown in Table 2.2-3 under the post development period. This table also illustrates where quality ratings can be predicted through measurement (△) and where the impact is felt if poor quality is recognized (X).

Table 2.2-3 Relationship of Quality Factors to Life-Cycle Phases

	Development					Eval	Post-Development			Benefit
Life-Cycle Phases / Factors	Reqmts Analysis	Prelim Design	Detail Design	Imple-mentation	Test & Integr	System Test & Instl	Operation	Revision	Transition	Expected Cost-Saved/Cost-to-Provide Ratio
Correctness	△	△	△	△	△	X	X	X		High
Reliability	△	△	△	△	△	X	X	X		High
Efficiency	△	△	△	△	△		X			Low
Integrity	△	△			△		X			Low
Usability	△	△			△	X	X	X		Medium
Survivability	△	△	△	△	△	X	X			Low
Maintainabilty	△	△	△	△	△			X	X	High
Verifiability	△	△	△	△	△	X		X	X	High
Flexibility	△	△	△	△	△			X	X	Medium
Portability	△	△	△	△					X	Medium
Reusability	△	△	△	△	△				X	Medium
Interoperability	△	△	△	△	△	X			X	Medium
Expandability	△	△	△	△	△				X	Medium

Legend: △ = When Quality is Measured and Predicted, X = When Impact of Poor Quality is Recognized

The size of this impact determines the cost savings that can be expected if a higher quality system is achieved through the application of the metrics. This cost savings is somewhat offset by the cost to apply the metrics and the cost to develop the higher quality software product as illustrated in Figure 2.2-1. The cost to apply the metrics is difficult to estimate for the first project in which they are applied. This is due to the training time for personnel applying metrics. Experience shows that a learning curve applies — that subsequent applications of metrics have a lower cost and greater cost saving opportunities.

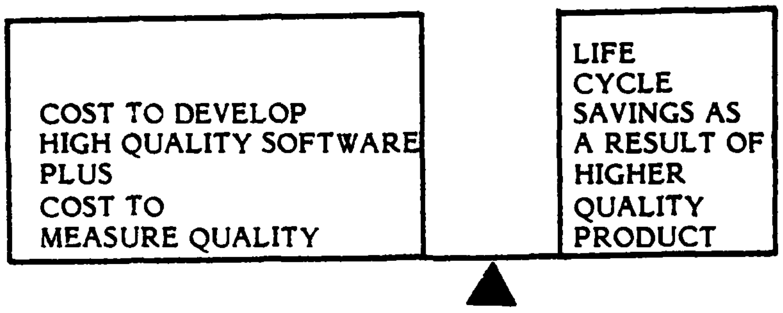


Figure 2.2-1 Benefit Tradeoff: Quality Costs vs Cost Savings

This cost to implement versus life-cycle cost reduction relationship exists for each quality factor. The benefit, cost-to-provide versus cost-saved ratio, for each factor is rated as high, medium, or low in the right hand column of Table 2.2-3. This relationship and the life-cycle implications of the quality factors should be considered when selecting the important factors for a specific system.

1c. Performance Tradeoffs Among the Quality Factors

As a result of steps 1a and 1b, a tentative list of quality factors should be produced. The next step is to consider the interrelationships among the factors selected. Tables 2.2-4 and 2.2-5 can be used as a guide for determining the relationships between the quality factors. Some factors are synergistic while other conflict. The impact of conflicting factors is that a lower quality level can be expected for each factor if both are required than can be expected if only one or the other is required. The synergistic (positive tradeoffs) and conflicts (negative tradeoffs) may reflect a more complex interrelationship of factors. For example, there may be a group of three factors which can all be enhanced together by a design decision. An effort should be made to identify such multiple tradeoffs for the particular software product.

Table 2.2-4 Relationships Between Software Quality Factors

LEGEND:
IF A HIGH DEGREE OF QUALITY IS PRESENT FOR ONE FACTOR, THE DEGREE OF QUALITY EXPECTED FOR THE OTHER FACTOR IS:

△ = HIGH
▲ = LOW

BLANK = NONE OR DEPENDENT UPON APPLICATION

SOFTWARE QUALITY FACTORS / SOFTWARE QUALITY FACTORS	CORRECTNESS	RELIABILITY	EFFICIENCY	INTEGRITY	USABILITY	SURVIVABILITY	MAINTAINABILITY	VERIFIABILITY	FLEXIBILITY	PORTABILITY	REUSABILITY	INTEROPERABILITY	EXPANDABILITY
CORRECTNESS													
RELIABILITY	△												
EFFICIENCY		▲											
INTEGRITY			▲										
USABILITY	△	△	▲										
SURVIVABILITY	△	△	▲	▲	△								
MAINTAINABILITY	△	△	▲		△								
VERIFIABILITY	△	△	▲	△	△		△						
FLEXIBILITY	△	▲	▲	▲	△	▲	△	△					
PORTABILITY			▲			▲			△				
REUSABILITY	△	▲	▲	▲	△	▲	△	△	△	△			
INTEROPERABILITY		▲	▲	▲		△	▲	▲	▲	△	△		
EXPANDABILITY	△	▲	▲	▲	△		△	△	△	△	△	△	

Table 2.2-5 Typical Factor Tradeoffs

RELIABILITY VS	EFFICIENCY	THE ADDITIONAL CODE REQUIRED TO PROVIDE ACCURACY AND TO PERFORM ANOMALY MANAGEMENT USUALLY INCREASES RUN TIME AND REQUIRES ADDITIONAL STORAGE.
	FLEXIBILITY REUSABILITY EXPANDABILITY INTEROPERABILITY	THE GENERALITY REQUIRED FOR FLEXIBLE, REUSABLE, AND EXPANDABLE SOFTWARE USUALLY INCREASES THE DIFFICULTY OF PROVIDING ACCURACY AND PERFORMING ANOMALY MANAGEMENT FOR SPECIFIC CASES.
EFFICIENCY VS	INTEGRITY	THE ADDITIONAL CODE AND PROCESSING REQUIRED TO CONTROL ACCESS TO CODE OR DATA USUALLY LENGTHENS RUN TIME AND REQUIRES ADDITIONAL STORAGE.
	USABILITY	THE ADDITIONAL CODE AND PROCESSING REQUIRED TO EASE AN OPERATOR'S TASK OR TO PROVIDE MORE USABLE OUTPUT USUALLY INCREASE RUNTIME AND REQUIRE ADDITIONAL STORAGE.
	SURVIVABILITY	THE ADDITIONAL CODE AND PROCESSING REQUIRED FOR MODULAR, RECONFIGURABLE, ANOMALY TOLERANT SOFTWARE RESULTS IN LESS EFFICIENT OPERATION.
	MAINTAINABILITY VERIFIABILITY	USING MODULAR, VISIBLE, SELF-DESCRIPTIVE CODE TO INCREASE MAINTAINABILITY AND VERIFIABILITY USUALLY INCREASES OVERHEAD AND RESULTS IN LESS EFFICIENT OPERATION. CODE WHICH IS OPTIMIZED FOR EFFICIENCY POSES PROBLEMS TO THE TESTER & MAINTAINER.
	FLEXIBILITY REUSABILITY	THE GENERALITY REQUIRED FOR FLEXIBLE AND REUSABLE SOFTWARE INCREASES OVERHEAD AND DECREASES EFFICIENCY.
	PORTABILITY	THE USE OF CODE OPTIMIZED FOR EFFICIENCY USUALLY DECREASES PORTABILITY.
	INTEROPERABILITY	THE OVERHEAD FOR CONVERSION FROM STANDARD DATA REPRESENTATIONS AND FOR THE USE OF STANDARD INTERFACE ROUTINES DECREASES OPERATING EFFICIENCY.
	EXPANDABILITY	THE USE OF MODULAR, GENERAL SOFTWARE USUALLY DECREASES OPERATING EFFICIENCY.
INTEGRITY VS	SURVIVABILITY	THE DISTRIBUTEDNESS REQUIRED FOR SURVIVABLE SOFTWARE INCREASES THE RISK OF UNAUTHORIZED ACCESS.
	FLEXIBILITY REUSABILITY	THE GENERALITY REQUIRED FOR FLEXIBLE AND REUSABLE SOFTWARE INCREASES THE RISK OF UNAUTHORIZED ACCESS.
	INTEROPERABILITY	COUPLED SYSTEM HAVE MORE AVENUES OF ACCESS, DIFFERENT USERS, AND COMMON DATA REPRESENTATIONS; THE Y OFTEN SHARE DATA AND CODE. THESE INCREASE THE POTENTIAL FOR ACCIDENTAL OR DELIBERATE ACCESS OF SENSITIVE DATA.
	EXPANDABILITY	THE GENERALITY REQUIRED FOR EXPANDABLE SOFTWARE INCREASES THE RISK OF UNAUTHORIZED ACCESS.
SURVIVABILITY VS	FLEXIBILITY PORTABILITY REUSABILITY	THE RECONFIGURABILITY REQUIRED FOR SURVIVABLE SOFTWARE REDUCES ITS FLEXIBILITY, PORTABILITY, AND REUSABILITY.
INTEROPERABILITY VS	MAINTAINABILITY VERIFIABILITY FLEXIBILITY	THE ADDITIONAL COMPLEXITY INTRODUCED BY COMMUNICATION, FUNCTIONAL INTERFACING, AND DATA COMMONALITY BETWEEN SYSTEMS INCREASES THE COMPLEXITY OF CHANGING, VERIFYING, AND MAINTAINING THE SOFTWARE

1d. Identify Most Important Quality Factors

Based on 1a through 1c, a list of quality factors considered to be important for the particular system can be compiled. The list should be organized in order of importance. A single decision maker can be assigned to choose the factors or the choice can be made by averaging several survey responses. The definitions of the factors chosen should be included with this list.

1e. Provide Explanation for Choice

The rationale for the decisions made during steps 1a through 1c should be documented. If a factor is not considered important for the system, a rationale may also be provided. For example, maintainability may not be emphasized because verifiability (given top priority) will ensure a thoroughly tested (and therefore highly maintainable) product.

2.2.2 An Example of Factors Specification

To illustrate the application of the above steps, consider a spare parts inventory control system. The inventory control system maintains inventory status and facilitates requisitioning, reordering, and issuing of spare parts to Air Force units in support of various missions. The planned life of the system is ten years.

Each step described previously will be performed with respect to the spare parts inventory control system.

1a. Consider Basic Characteristics of the Application

Utilizing Table 2.2-2 and considering the unique characteristics of the spare parts inventory control system resulted in the following:

Characteristic	Related Quality Factor
Critical Support for a Flying Unit	Reliability Correctness Verifiability Survivability

Characteristic	Related Quality Factor
Long Life Cycle With Stable Hardware And Software Requirements	Maintainability
Utilized By Air Force Maintenance Personnel	Usability
Interfaces with other Air Force Inventory Systems (e.g. Supplies)	Interoperability

1b. Consider Life Cycle Implications

For the five quality factors identified in 1a, determine the life cycle cost benefits according to Table 2.2-3.

QUALITY FACTORS	COST BENEFIT RATIO
Reliability	High
Correctness	High
Verifiability	High
Survivability	Low
Maintainability	High
Usability	Medium
Interoperability	Medium

1c. Perform Trade Offs Among Quality Factors

Using Table 2.2-4, there are no conflicts which need to be considered.

1d. Identify Most Important Quality Factors

Using Table 2.2-1 and the guidance provided by steps 1a through 1c, the following factors are identified in order of importance; provide the definitions.

CORRECTNESS	-Extent to which the software satisfies its specifications and fulfills the user's mission objectives.

RELIABILITY	-Probability that the software will perform its logical operations in the specified environment without failure.
USABILITY	-Effort for training and software operation -familiarization, input preparation, execution, output interpretation.
VERIFIABILITY	-Effort to verify the specified software operation and performance.
SURVIVABILITY	-Probability that the software will continue to perform or support critical functions when a portion of the system is inoperable.
MAINTAINABILITY	-Average effort to locate and fix a software failure.
INTEROPERABILITY	-Effort to couple the software of one system to the software of another system.

1e. <u>Provide Explanation for Choice</u>
Document the rationales for the decisions made in the above step.

CORRECTNESS	-System performs critical spare . parts provision function.
RELIABILITY	-System performs critical spare parts provision functions in field environment.
VERIFIABILITY	-System performs critical spare parts provision functions.
SURVIVABILITY	-System performs critical spare parts provision function in field environment and will interface with other systems.

USABILITY	-System will be used by military personnel with minimum computer training.
MAINTAINABILITY	-System life cycle is projected to be 10 years and will operate in the field and be maintained by military personnel.
INTEROPERABILITY	-System will interface with other inventory systems.

2.3 IDENTIFYING CRITICAL SOFTWARE ATTRIBUTES

2.3.1 Procedures

The second level of identifying the quality requirements involves proceeding from the user-oriented quality factors to the software-oriented criteria. Sets of criteria, which are attributes of the software, are related to the various factors by definition. Their identification is automatic and represents a more detailed specification of the quality requirements. Identification of a quality factor does not automatically mean that all criteria within that factor are equally important. Tradeoffs and synergisms may exist between criteria within the same factor. A subset of the criteria within a factor may be identified.

2a. Identify Critical Software Attributes Required

Table 2.3-1 is used to identify the software attributes (criteria) associated with the chosen software quality factors.

Table 2.3-1 Software Criteria and Related Quality Factors

QUALITY FACTOR	SOFTWARE CRITERIA
CORRECTNESS	COMPLETENESS CONSISTENCY SIMPLICITY SPECIFICITY TRACEABILITY
EFFICIENCY	EFFECTIVENESS
FLEXIBILITY	GENERALITY MODULARITY SELF-DESCRIPTIVENESS SIMPLICITY
INTEGRITY	SYSTEM ACCESSIBILITY VIRTUALITY
INTEROPERABILITY	AUGMENTABILITY COMMONALITY COMMUNICATIVENESS FUNCTIONAL OVERLAP INDEPENDENCE MODULARITY SYSTEM COMPATIBILITY
MAINTAINABILITY	CONCISENESS CONSISTENCY MODULARITY SELF-DESCRIPTIVENESS SIMPLICITY VISIBILITY

Table 2.3-1 (continued)

QUALITY FACTOR	SOFTWARE CRITERIA
EXPANDABILITY	AUGMENTABILITY GENERALITY MODULARITY SIMPLICITY SPECIFICITY VIRTUALITY
PORTABILITY	INDEPENDENCE MODULARITY SELF-DESCRIPTIVENESS
RELIABILITY	ACCURACY ANOMALY MANAGEMENT CONSISTENCY SIMPLICITY
REUSABILITY	APPLICATION INDEPENDENCE DOCUMENT ACCESSIBILITY FUNCTIONAL SCOPE GENERALITY INDEPENDENCE MODULARITY SELF DESCRIPTIVENESS SIMPLICITY SYSTEM CLARITY
VERIFIABILITY	MODULARITY SELF-DESCRIPTIVENESS SIMPLICITY SPECIFICITY VISIBILITY

Table 2.3-1 (continued)

QUALITY FACTOR	SOFTWARE CRITERIA
USABILITY	COMMUNICATIVENESS OPERABILITY TRAINING VIRTUALITY VISIBILITY
SURVIVABILITY	ANOMALY MANAGEMENT AUTONOMY DISTRIBUTEDNESS MODULARITY RECONFIGURABILITY

2b. <u>Provide Definitions</u>

Table 2.3-2 should be used to provide the definitions of criteria as part of the specification.

Table 2.3-2 Definitions of Software Criteria

SOFTWARE CRITERION	DEFINITION
ACCURACY	Those attributes of the software which provide the required precision in calculations and outputs.
ANOMALY MANAGEMENT	Those attributes of the software which provide for continuity of operations under, and recovery from nonnominal conditions.
APPLICATION INDEPENDENCE	Attributes of the software which determine its dependency on the software application (database system, data structure, system libraries routines, microcode, computer architecture and algorithms)
AUGMENTABILITY	Those attributes of the software which provide for expansion of capability for functions and data.
AUTONOMY	Those attributes of the software which determine its nondependency on interfaces and functions.
COMMONALITY	Those attributes of the software which provide for the use of interface standard for protocols, routines, and data representations.
COMMUNICATIVENESS	Those attributes of the software which provide useful inputs and outputs which can be assimilated.
COMPLETENESS	Those attributes of the software which provide full implementation of the functions required.
CONCISENESS	Those attributes of the software which provide for implementation of a function with a minimum amount of code.
CONSISTENCY	Those attributes of the software which provide for uniform design and implementation techniques and notation.
DISTRIBUTEDNESS	Those attributes of the software which determine the degree to which software functions are geographically or logically separated within the system.
DOCUMENT ACCESSIBILITY	Attributes of the software which provide easy access to and selective use of system components.

Table 2.3-2 (continued)

SOFTWARE CRITERION	DEFINITION
EFFECTIVENESS	Those attributes of the software which provide for minimum utilization of resources (processing time, storage, operator time) in performing functions.
FUNCTIONAL OVERLAP	A comparison between two systems to determine the number of functions common to both systems.
FUNCTIONAL SCOPE	Those attributes of the software which provide the scope of functions required to be performed i.e. specificity, commonality and completeness.
GENERALITY	Those attributes of the software which provide breadth to the functions performed with respect to the application.
INDEPENDENCE	Those attributes of the software which determine its non-dependency on the software environment (computing system, operating system, utilities, input/output routines, libraries).
MODULARITY	Those attributes of the software which provide a structure of highly cohesive modules with optimum coupling.
OPERABILITY	Those attributes of the software which determine operations and procedures concerned with the operation of the software.
RECONFIGURABILITY	Those attributes of the software which provide for continuity of system operation when one or more processors, storage units, or communication links fail.
SELF-DESCRIPTIVENESS	Those attributes of the software which provide explanation of the implementation of a function.
SIMPLICITY	Those attributes of the software which provide for the definition and implementation of functions in the most non-complex and understandable manner.
SPECIFICITY	Those attributes of the software which provide for singularity in the definition and implementation of functions.
SYSTEM ACCESSIBILITY	Those attributes of the software which provide for control and audit of access of software and data.

Table 2.3-2 (continued)

SOFTWARE CRITERION	DEFINITION
SYSTEM CLARITY	Those attributes of the software which provide clear description of program structure in the most non-complex, easily understandable and modifiable manner.
SYSTEM COMPATIBILITY	A measure of the hardware, software and communication compatibility of two systems.
TRACEABILITY	Those attributes of the software which provide a thread of origin from the implementation to the requirements with respect to the specific development envelope and operational environment.
TRAINING	Those attributes of the software which provide transition from current operation or provide initial familiarization.
VIRTUALITY	Those attributes of the software which present a system that does not require user knowledge of the physical, logical, or topological characteristics (e.g., number of processors/disks, storage locations).
VISIBILITY	Those attributes of the software which provide status monitoring of the development and operation (e.g., instrumentation).

2.3.2 Example of Identifying Software Criteria

Continuing with the example of paragraph 2.2.2, the software criteria for the identified quality factors would be chosen.

2a. Identify Critical Software Attributes

Using the relationships provided in Table 2.3-1, the software criteria shown in Table 2.3-3 would be identified. Evaluation of the definitions of the criteria in the context of the software product and its quality goals, may allow a number of the resulting criteria to be eliminated.

Table 2.3-3 Software Criteria to Factor Relationships

SOFTWARE CRITERIA	RELATED FACTOR						
	CO	RL	SV	MA	VE	US	IP
TRACEABILITY	X						
CONSISTENCY	X	X		X			
COMPLETENESS	X						
ANOMALY MANAGE-MENT	X		X				
ACCURACY		X					
SIMPLICITY		X		X	X		
CONCISENESS				X			
MODULARITY			X	X	X		X
SELF-DESCRIPTIVENESS				X	X		
OPERABILITY						X	
TRAINING						X	
COMMUNICATIVENESS						X	
COMMONALITY							X
FUNCTIONAL OVERLAP							X
INDEPENDENCE							X
SYSTEM COMPATIBILITY							X
VISIBILITY				X	X	X	
AUGMENTABILITY							X
MODULARITY							X
AUTONOMY			X				
DISTRIBUTEDNESS			X				
RECONFIGURABILITY			X				
SPECIFICITY					X		

CO = Correctness, RL = Reliability, SV = Survivability
MA = Maintainability, VE = Verifiability,
US = Usability, IP = Interoperability

2b. Provide Definitions

The definitions for each of these software criteria, as shown in Table 2.3-2 would also be provided as part of the specification.

2.4 ESTABLISHING QUANTIFIABLE GOALS

2.4.1 Procedures

The third and last level, which is the most detailed and quantified, requires precise statements of the level of quality that will be acceptable for the software product.

Currently, the underlying mathematical relationships which allow measurement at this level of precision do not exist for all of the quality factors. The mechanism for making the precise statement for any quality factor is a rating or figure-of-merit of the factor. The underlying basis for the ratings of all factors except reliability and survivability is the effort or cost required to perform a function such as to correct or modify the design or program. For example, rating for maintainability might be that the average time to fix a problem should be five man-days or that 90% of the problem fixes should take less than six man-days. This rating would be specified as a quality requirement. To comply with this specification, the software would have to exhibit characteristics which, when present, give an indication that the software will perform to this rating. These characteristics are measured by metrics which are inserted into a mathematical relationship to obtain the predicted rating. Note that the reliability ratings are provided in terms familiar to traditional hardware reliability. Just as in hardware reliability there are significant differences between ratings of .9 and .99.

In order to choose ratings such as the two mentioned above, data must be available which allows the decision maker to know what is a "good rating" or perhaps what is the industry average. Currently there is generally a lack of good historical data to establish these expected levels of operations and maintenance performance for software. There are significant efforts underway to compile historical data and derive the associated performance statistics (DUVA76). Individual software development organizations and System Program Offices should attempt to compile historical data for their particular environment. Any environment-unique data available should be used as a check against the data provided as guidelines in this manual. The data utilized in this section is based on experiences applying the metrics to several large command and control software systems and other experiences reported in the literature.

3a. Specify Rating for Each Quality Factor

After identification of the critical quality factors, specific performance levels or ratings required for each factor should be specified. Tables 2.4-1 and 2.4-2 should be used as a guideline for identifying the ratings for the particular factors. Note that mathematical relationships have not been established for some of the factors. In those cases, it is advisable not to levy requirements for meeting a specific quality rating but instead specify the relative importance (priority) of the quality factor as a development goal.

3b. Identify Specific Metrics to be Applied

The next step or an alternative to 3a is to identify the specific metrics which will be applied to the various software products produced during the development. The Metric Worksheets described in Appendix A can be used for this purpose or Table 2.4-3 can be used to identify the metrics and reference can be made to Appendix C where definitions of the metrics are provided. Detailed examination may allow a subset of the metrics within a criteria to be isolated.

3c. *Specification of Metric Threshold Values*

In lieu of specifying quality ratings or in addition to the ratings, specific minimum values for particular metrics may be specified. This technique is equivalent to establishing a standard which is to be adhered to. Measurements less than the value established are to be reported. Typical values can be derived by applying the metrics to software products developed in a particular environment or by looking at the scores reported in (MCCA77), (MCCA80) or Volume 1 of this report. When establishing these threshold values based on past project data, projects which have been considered successful, i.e., have demonstrated good characteristics during their life cycle should be chosen. For example, a system which has been relatively cost-effective to maintain over its operational history should be chosen and the metrics related to maintainability applied to establish threshold values. Incentives may also be offered if a particular metric exceeds a maximum threshold value.

Table 2.4-1 Quality Factor Ratings

QUALITY FACTOR	RATING EXPLANATION	RATING GUIDELINES				
RELIABILITY*	Rating is in terms of the number of errors that occur after the start of formal testing. Rating = 1-$\frac{\text{Number of Errors}}{\text{Number of Lines of source code excluding comments}}$	RATING	.9	.98**	.99	.999
		$\frac{\text{ERRORS}}{\text{100 LOC}}$	10	2	1	.1
MAINTAINA-BILITY*	Rating is in terms of the average amount of effort required to locate and fix an error in an operational program. Rating = 1-.1 (Average number of man days per fix)	RATING	.3	.5	.7**	.9
		AVERAGE EFFORT (MAN DAYS)	7	5	3	1
PORTABILITY*	Rating is in terms of the effort required to convert a program to run in another environment with respect to the effort required to originally implement the program. Rating = 1-$\frac{\text{Effort to Transport}}{\text{Effort to Implement}}$	RATING	.25	.5**	.75	.9
		% OF ORIGINAL EFFORT	75	50	25	10
FLEXIBILITY*	Rating is in terms of the average effort required to extend a program to include other requirements. Rating = 1-.05(Average number of man days to change)	RATING	.3	.5**	.7	.9
		AVERAGE EFFORT (MAN DAYS)	14	10	6	2
REUSABILITY*	Rating is in terms of the effort required to convert a program to a different application with respect to the effort required to build a new program. Rating = 1-$\frac{\text{Effort to Convert}}{\text{Effort to Build}}$	RATING	.2	.4**	.75	.9
		% OF EFFORT TO BUILD	80	60	25	10

Table 2.4-1 (Continued)

QUALITY FACTOR	RATING EXPLANATION	RATING GUIDELINES				
INTEROPERA-BILITY*	Rating is in terms of the effort required to couple the system to another system. Rating = $1 - \frac{\text{Effort to Modify}}{\text{Effort to Build}}$	RATING	.2	.5	.75	.9
		% OF EFFORT TO BUILD	80	50	25	10
EXPANDABILITY*	Rating is in terms of the effort to increase software capability, performance and original development effort.	RATING	.4	.5	.6	.7
		% OF EFFORT TO DEVELOP	60	45	30	10

NOTES

* Data collected to date provides some basis upon which to allow quantitative ratings for these quality factors. These ratings should be modified based on data collected within a specific development environment. Data has not been collected to support ratings of the other quality factors.

** Indicates rating which might be considered current industry average.

Table 2.4-2 Quality Factor Rating Explanation

QUALITY FACTOR	RATING EXPLANATION (Guidelines Not Established)
CORRECTNESS	The function which the software is to perform is incorrect. The rating is in terms of effort required to implement the correct function.
EFFICIENCY	The software does not meet performance (speed, storage) requirements. The rating is in terms of effort required to modify software to meet performance requirements.
INTEGRITY	The software does not provide required security. The rating is in terms of effort required to implement proper levels of security.
USABILITY	There is a problem related to operation of the software, the user interface, or the input/output. The rating is in terms of effort required to improve human factors to acceptable level.
VERIFIABILITY	The rating is in terms of effort required to test changes or fixes.
SURVIVABILITY	The rating is in terms of the number of survivability related errors that occur after the start of formal testing.

Table 2.4-3 Quality Metrics Related to Factors

QUALITY FACTOR	METRICS	ACRONYM*
CORRECTNESS	COMPLETENESS CHECKLIST	CP.1
	PROCEDURE CONSISTENCY MEASURE	CS.1
	DATA CONSISTENCY MEASURE	CS.2
	DESIGN STRUCTURE MEASURE	SI.1
	STRUCTURED LANGUAGE OR PREPROCESSOR	SI.2
	DATA AND CONTROL FLOW COMPLEXITY MEASURE	SI.3
	CODING SIMPLICITY MEASURE	SI.4
	SCOPE OF FUNCTION MEASURE	SP.1
	CROSS REFERENCE	TR.1
RELIABILITY	ERROR TOLERANCE/CONTROL CHECKLISTS	AM.1
	IMPROPER INPUT DATA CHECKLIST	AM.2
	COMPUTATIONAL FAILURES CHECKLIST	AM.3
	HARDWARE FAULTS CHECKLIST	AM.4
	DEVICE ERROR CHECKLIST	AM.5
	COMMUNICATION ERRORS CHECKLIST	AM.6
	NODE/COMMUNICATIONS FAILURES	AM.7
	ACCURACY CHECKLIST	AY.1
	PROCEDURE CONSISTENCY MEASURE	CS.1
	DATA CONSISTENCY MEASURE	CS.2
	DESIGN STRUCTURE MEASURE	SI.1
	STRUCTURED LANGUAGE OR PREPROCESSOR	SI.2
	DATA AND CONTROL FLOW COMPLEXITY MEASURE	SI.3
	CODING SIMPLICITY MEASURE	SI.4
EFFICIENCY	PERFORMANCE REQUIREMENTS	EF.1
	ITERATIVE PROCESSING EFFICIENCY MEASURE	EF.2
	DATA USAGE EFFICIENCY MEASURE	EF.3
	STORAGE EFFICIENCY MEASURE	EF.4

*Acronym references relate to definitions in Appendix C

Table 2.4-3 (Continued)

QUALITY FACTOR	METRICS	ACRONYM*
INTEGRITY	ACCESS CONTROL CHECKLIST	SA.1
	ACCESS AUDIT CHECKLIST	SA.2
	SYSTEM/DATA INDEPENDENCE CHECKLIST	VR.1
USABILITY	USER INPUT INTERFACE MEASURE	CM.1
	USER OUTPUT INTERFACE MEASURE	CM.2
	OPERABILITY CHECKLIST	OP.1
	TRAINING CHECKLIST	TN.1
	SYSTEM/DATA INDEPENDENCE CHECKLIST	VR.1
	MODULE TESTING MEASURE	VS.1
	INTEGRATION TESTING MEASURE	VS.2
	SYSTEM TESTING MEASURE	VS.3
SURVIVABILITY	ERROR TOLERANCE/CONTROL CHECKLIST	AM.1
	IMPROPER INPUT DATA CHECKLIST	AM.2
	COMPUTATIONAL FAILURES CHECKLIST	AM.3
	HARDWARE FAULTS CHECKLIST	AM.4
	DEVICE ERRORS CHECKLIST	AM.5
	COMMUNICATION ERRORS CHECKLIST	AM.6
	NODE/COMMUNICATIONS FAILURES CHECKLIST	AM.7
	INTERFACE COMPLEXITY MEASURE	AU.1
	SELF-SUFFICIENCY CHECKLIST	AU.2
	DESIGN STRUCTURE CHECKLIST	DI.1
	MODULAR IMPLEMENTATION MEASURE	MO.2
	MODULAR DESIGN MEASURE	MO.3
	RESTRUCTURE CHECKLIST	RE.1
MAINTAINABILITY	HALSTEAD'S MEASURE	CO.1
	PROCEDURE CONSISTENCY MEASURE	CS.1

*Acronym references relate to definitions in Appendix C

Table 2.4-3 (Continued)

QUALITY FACTOR	METRICS	ACRONYM*
MAINTAINABILITY (continued)	DATA CONSISTENCY MEASURE	CS.2
	MODULAR IMPLEMENTATION MEASURE	MO.2
	MODULAR DESIGN MEASURE	MO.3
	QUANTITY OF COMMENTS	SD.1
	EFFECTIVENESS OF COMMENTS MEASURE	SD.2
	DESCRIPTIVENESS OF LANGUAGE MEASURE	SD.3
	DESIGN STRUCTURE MEASURE	SI.1
	STRUCTURED LANGUAGE OR PREPROCESSOR	SI.2
	DATA AND CONTROL FLOW COMPLEXITY MEASURE	SI.3
	CODING SIMPLICITY MEASURE	SI.4
	MODULE TESTING MEASURE	VS.1
	INTEGRATION TESTING MEASURE	VS.2
	SYSTEM TESTING MEASURE	VS.3
VERIFIABILITY	MODULAR IMPLEMENTATION MEASURE	MO.2
	MODULAR DESIGN MEASURE	MO.3
	QUANTITY OF COMMENTS	SD.1
	EFFECTIVENESS OF COMMENTS MEASURE	SD.2
	DESCRIPTIVENESS OF IMPLEMENTATION LANGUAGE MEASURE	SD.3
	DESIGN STRUCTURE MEASURE	SI.1
	STRUCTURED LANGUAGE OR PREPROCESSOR	SI.2
	DATA AND CONTROL FLOW COMPLEXITY MEASURE	SI.3
	CODING SIMPLICITY MEASURE	SI.4
	SCOPE OF FUNCTION MEASURE	SP.1
	MODULE TESTING MEASURE	VS.1

*Acronym references relate to definitions in Appendix C

Table 2.4-3 (Continued)

QUALITY FACTOR	METRICS	ACRONYM*
VERIFIABILITY (continued)	INTEGRATION TESTING MEASURE	VS.2
	SYSTEM TESTING MEASURE	VS.3
FLEXIBILITY	MODULE REFERENCE BY OTHER MODULES	GE.1
	IMPLEMENTATION FOR GENERALITY CHECKLIST	GE.2
	MODULAR IMPLEMENTATION MEASURE	MO.2
	MODULAR DESIGN MEASURE	MO.3
	QUANTITY OF COMMENTS	SD.1
	EFFECTIVENESS OF COMMENTS MEASURE	SD.2
	DESCRIPTIVENESS OF LANGUAGE MEASURE	SD.3
	DESIGN STRUCTURE MEASURE	SI.1
	STRUCTURED LANGUAGE OR PREPROCESSOR	SI.2
	DATA AND CONTROL FLOW COMPLEXITY MEASURE	SI.3
	CODING SIMPLICITY MEASURE	SI.4
PORTABILITY	SOFTWARE SYSTEM INDEPENDENCE MEASURE	ID.1
	MACHINE INDEPENDENCE MEASURE	ID.2
	MODULAR IMPLEMENTATION MEASURE	MO.2
	MODULAR DESIGN MEASURE	MO.3
	QUANTITY OF COMMENTS	SD.1
	EFFECTIVENESS OF COMMENTS MEASURE	SD.2
	DESCRIPTIVENESS OF LANGUAGE MEASURE	SD.3
REUSABILITY	DATA BASE SYSTEM INDEPENDENCE	AI.1
	DATA STRUCTURE	AI.2
	ARCHITECTURE STANDARDIZATION	AI.3
	MICROCODE INDEPENDENCE	AI.4
	ALGORITHM	AI.5
	ACCESS NO-CONTROL	DA.1

*Acronym references relate to definitions in Appendix C

Table 2.4-3 (Continued)

QUALITY FACTOR	METRICS	ACRONYM*
REUSABILITY (continued)	WELL-STRUCTURED DOCUMENTATION	DA.2
	SELECTIVE USABILITY	DA.3
	FUNCTION SPECIFICITY	FS.1
	FUNCTION COMMONALITY	FS.2
	FUNCTION COMPLETENESS	FS.3
	MODULE REFERENCE BY OTHER MODULES	GE.1
	IMPLEMENTATION FOR GENERALITY CHECKLIST	GE.2
	SOFTWARE SYSTEM INDEPENDENCE	ID.1
	MACHINE INDEPENDENCE	ID.2
	MODULAR IMPLEMENTATION MEASURE	MO.2
	MODULAR DESIGN MEASURE	MO.3
	INTERFACE COMPLEXITY	SC.1
	PROGRAM FLOW COMPLEXITY	SC.2
	APPLICATION FUNCTIONAL COMPLEXITY	SC.3
	COMMUNICATION COMPLEXITY	SC.4
	STRUCTURE CLARITY	SC.5
	QUANTITY OF COMMENTS	SD.1
	EFFECTIVENESS OF COMMENTS MEASURE	SD.2
	DESCRIPTIVENESS OF LANGUAGE MEASURE	SD.3
	DESIGN STRUCTURE MEASURE	SI.1
	STRUCTURED LANGUAGE OR PREPROCESSOR	SI.2
	DATA AND CONTROL FLOW COMPLEXITY MEASURE	SI.3
	CODING SIMPLICITY MEASURE	SI.4
INTEROPERABI-LITY	DATA STORAGE EXPANSION MEASURE	AG.1
	COMPUTATIONAL EXTENSIBILITY MEASURE	AG.2
	CHANNEL EXTENSIBILITY MEASURE	AG.3
	DESIGN EXTENSIBILITY CHECKLIST	AG.4

*Acronym references relate to definitions in Appendix C

Table 2.4-3 (Continued)

QUALITY FACTOR	METRICS	ACRONYM*
INTEROPERABILITY (continued)	COMMUNICATION COMMONALITY CHECKLIST	CL.1
	DATA COMMONALITY CHECKLIST	CL.2
	COMMON VOCABULARY CHECKLIST	CL.3
	USER INPUT INTERFACE MEASURE	CM.1
	USER OUTPUT INTERFACE MEASURE	CM.2
	FUNCTIONAL OVERLAP MEASURE	FO.1
	SOFTWARE SYSTEM INDEPENDENCE MEASURE	ID.1
	MACHINE INDEPENDENCE MEASURE	ID.2
	MODULAR IMPLEMENTATION MEASURE	MO.2
	MODULAR DESIGN MEASURE	MO.3
	COMMUNICATIONS COMPATIBILITY CHECKLIST	SY.1
	DATA COMPATIBILITY CHECKLIST	SY.2
	HARDWARE COMPATIBILITY CHECKLIST	SY.3
	SOFTWARE COMPATIBILITY CHECKLIST	SY.4
	DOCUMENTATION FOR OTHER SYSTEM	SY.5
EXPANDABILITY	DATA STORAGE EXPANSION MEASURE	AG.1
	COMPUTATION EXTENSIBILITY MEASURE	AG.2
	CHANNEL EXTENSIBILITY MEASURE	AG.3
	DESIGN EXTENSIBILITY CHECKLIST	AG.4
	MODULE REFERENCE BY OTHER MODULES	GE.1
	IMPLEMENTATION FOR GENERALITY CHECKLIST	GE.2
	MODULAR IMPLEMENTATION MEASURE	MO.2
	MODULAR DESIGN MEASURE	MO.3
	DESIGN STRUCTURE MEASURE	SI.1
	STRUCTURED LANGUAGE OR PREPROCESSOR	SI.2
	DATA AND CONTROL FLOW COMPLEXITY	SI.3
	CODING SIMPLICITY MEASURE	SI.4
	SCOPE OF FUNCTION MEASURE	SP.1
	SYSTEM/DATA INDEPENDENCE CHECKLIST	VR.1

*Acronym references relate to definitions in Appendix C

2.4.2 Example of Metrics

Using the example of paragraph 2.2.2, the quality ratings would be specified as follows.

3a. Specific Quality Factor Ratings

Ratings for two of the five important quality factors can be established using Table 2.4-1.

Reliability	.99	Require less than one error per 100 lines of code to be detected during formal testing.
Maintainability	.8	Require less than or equal to 2 man days as an average level of maintenance for correcting an error.

These ratings can also be established at each measurement period (see Table 3.1-1) during the software development process as follows:

QUALITY FACTOR	MEASUREMENT PERIODS				
	REQ	PDR	CDR	IMPL	ACCEPT
Reliability	.8	.8	.9	.9	.99
Maintainability	.7	.7	.8	.8	.8

The progressively better scores are required because there is more detailed information in the later phases of the development to which to apply the metrics and more confidence in the metrics' indication of quality. This is analagous to the concept of reliability growth. For other quality factors see step 3b.

3b. Identify Specific Metrics to be Applied

The metrics to be applied to assess the level of each important quality factor are chosen from Table 2.4-3. A subset is shown in Table 2.4-4.

Table 2.4-4 Software Metric to Factor Relationship-Subset

METRIC	QUALITY FACTOR				
	Reliability	Maintainability	Correctness	Usability	Interoperability
Accuracy Checklist	X				
Error Tolerance Checklist	X				
.					
Complexity Measure	X	X			
Coding Simplicity Measure	X	X			
Modular Implementation Measure		X			
.					
Quantity of Comments		X			
Effectiveness of Comments		X			
.					
Cross Reference Checklist			X		
Completeness Checklist			X		
Halstead's Measure		X			
Data Consistency Measure		X	X		
.					
User Input Interface Measure				X	X
Communications Commonality					X
Data Commonality Checklist					X
.					
Documentation for Other Systems					X

3c. Specify Threshold Values

The following threshold values are established based on past experience and to provide a goal for the quality factors that were not given ratings. They were derived by determining the average scores of past applications of the metrics.

Cross Reference Checklist	.9
Completeness Checklist	1.0
Halstead's Measure	.9
Data Consistency Measure	.6
Training Checklist	.75
User Input Interface Measure	.75
User Output Interface Measure	.75
Communications Commonality	.8
Data Commonality Checklist	.8

2.5 EVALUATION OF DEVELOPMENT PLAN

In an acquisition environment the initial benefits of utilizing the quality metrics concepts are realized in the source selection process. The acquisition office should include the quality goals established as software requirements in the Request for Proposal. The software attributes should also be identified as required characteristics in the software and the metrics established as the vehicles for assessing their existence. The bidders should be required to describe how they plan to provide those characteristics in the software. This discussion should be provided in the portion of the proposal that describes their development plan.

The description of the bidders approach for including the required attributes in the software not only forces acknowledgement of these additional requirements but also provides additional information with which to evaluate the bidders during source selection.

SECTION 3
APPLYING METRICS

3.1 WHEN TO TAKE MEASUREMENTS

The software quality metrics are oriented toward the availability of information about the software system as it progresses in its development. In the early phases of the development, the metrics are applied to the documentation produced to describe the concepts of the system and its design. In the later phases the metrics are oriented not only to documentation but also to the source code that is available.

Thus, the application of the metrics logically follows the phased development of software. The first application of the metric is at the end of the requirements analysis phase. The next application is during design. If the design phase has been decomposed into a preliminary design phase and a detailed design phase, the metrics should be applied at the end of each of those phases. During implementation, i.e., coding, the metrics oriented toward the source code should be applied periodically to assess the quality growth exhibited as the code evolves. The timing of the application of the metrics is shown in Figure 3.1-1. The application of the metrics can be done during or just prior to formal customer reviews (as shown in Figure 3.1-1) or during equivalent activities conducted by the development personnel.

In the case of reusable software, metrics may already exist from being applied during a previous project. Other metrics may change when re-evaluated later in the life-cycle, e.g., during maintenance. Maintainability, reliability and expandability factors may be re-evaluated as maintenance and upgrade activities occurred for fielded systems.

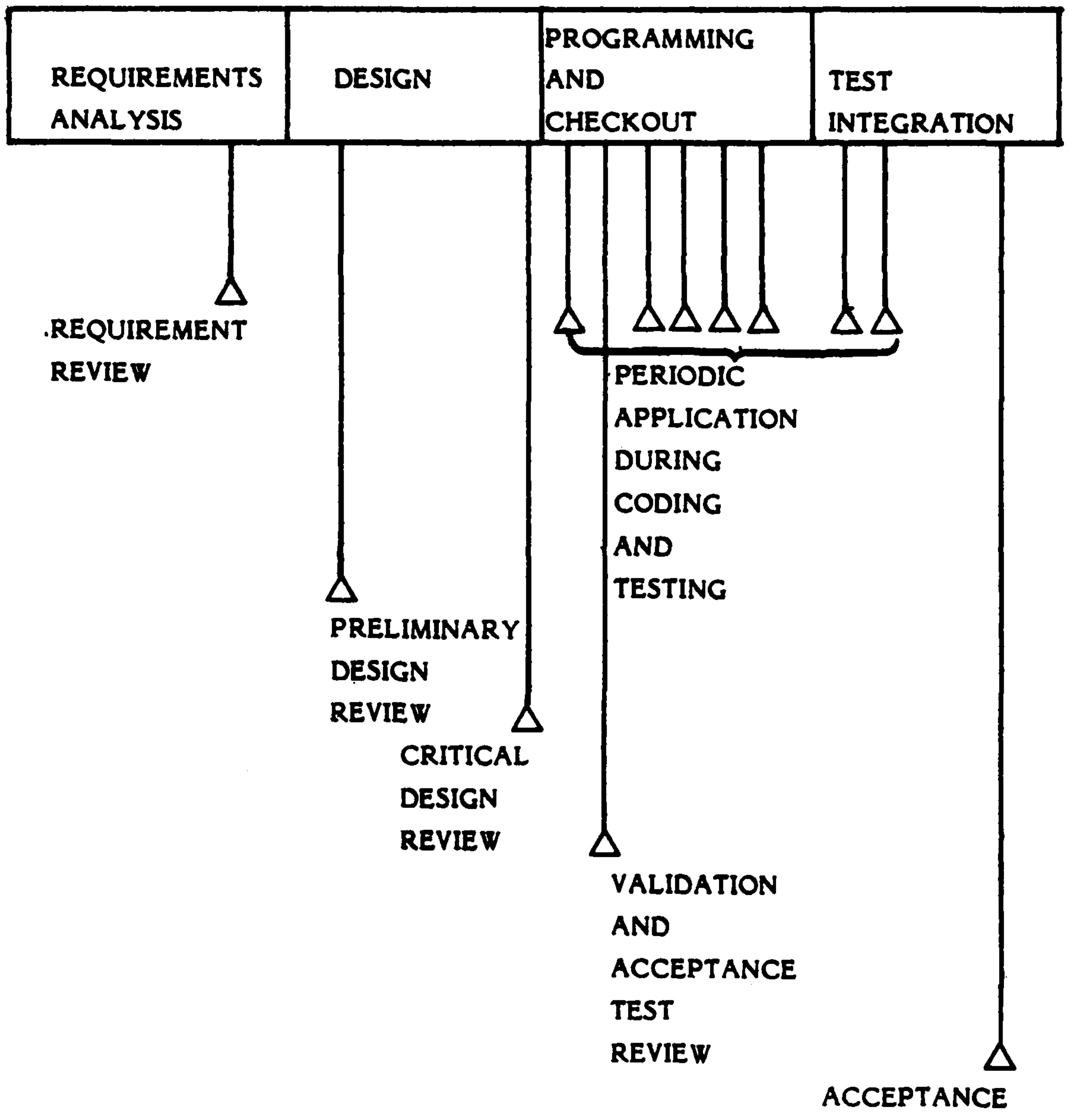


Figure 3.1-1 Timing of Metrics Application

3.2 SOURCES OF QUALITY INFORMATION

A typical minimum set of documents and source code are shown in Figure 3.2-1. These documents plus the source code are the sources of the metrics information used to derive the quality ratings.

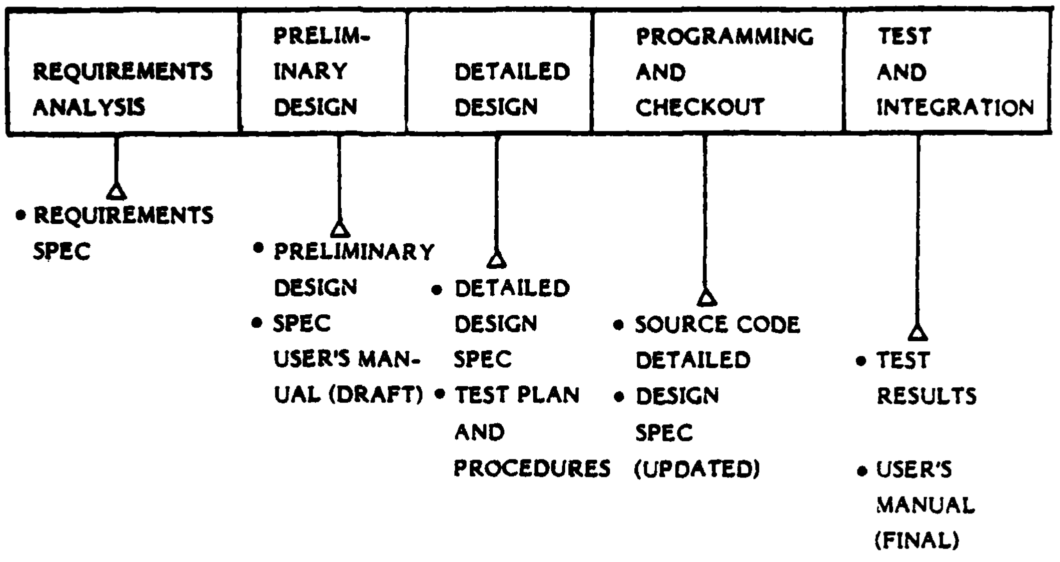


Figure 3.2-1: Sources of Quality Metric Data

3.3 APPLICATION OF THE METRICS

Application of the metrics can be accomplished by using: the metric worksheet contained in Appendix A for gathering data, the metric tables in Appendix B to translate the measurements into metric scores and the data in Appendix C for definitions and interpretations of individual metrics.

The metric worksheets are organized as follows. In the header portion of the worksheet is the information which (1) identifies the phase during which the worksheet is initially used and the level (system or module) to which the worksheet applies, (2) identifies the system and the module to which the worksheet has been applied, and (3) identifies the date and the inspector who took the measurements. The remaining portion of each worksheet contains the measurements to be taken and questions to be answered. These measurements and questions are organized by quality factors identified in parentheses. Each logical group of measurements and questions have a group identifier and group number. Each question contains a reference to the applicable metric.

When applying the measurements, only those measurements and questions that relate to the quality factors chosen as quality goals should be applied. A complete metric worksheet correlation matrix is shown in Table 3.3.1. The metric worksheet correlation matrix provides a quality factor to metric relationship. It also provides an individual metric to metric worksheet relationship.

Metric Worksheet #1 and #2 contain system level metrics and are applied at the system or major subsystem (CPCI) level to the System Requirements Specification, the Preliminary Design Specification, the User's Manual, and the Test documentation. Metric Worksheets #3 and #4 contain module level metrics and are applied to each module's design (Detailed Design Specification) and implementation (source code).

The metric tables in Appendix B are utilized to translate the raw data from the metric worksheets into individual metric scores. The metric tables in Appendix B are listed alphabetically by quality criteria. The metric tables are arranged as follows. In the header portion of the table is a reference to the quality criteria and the quality factors. The body of the table contains the instructions for computing individual metric scores with a reference to the metric worksheet that the raw data may be obtained from.

Table 3.3-1 METRIC WORKSHEET CORRELATION

QUALITY FACTORS														PHASES				
CORRECTNESS	RELIABILITY	EFFICIENCY	INTEGRITY	USABILITY	SURVIVABILITY	MAINTAINABILITY	VERIFIABILITY	FLEXIBILITY	PORTABILITY	REUSABILITY	INTEROPERABILITY	EXPANDABILITY	CRITERIA/ METRIC	REQUIREMENTS ANALYSIS	PRELIMINARY DESIGN	DETAILED DESIGN	IMPLEMENTATION	TEST & INTEGRATION
	X												ACCURACY					
													AY.1	1.2	2.2	3.2	(3.2),4.11	
	X				X								ANOMALY MANAGE-MENT					
													AM.1		2.2	3.2	4.2	2.2
													AM.2	1.2		3.2	4.7	
													AM.3	1.2		3.2	4.2	
													AM.4	1.2	2.2			2.2
													AM.5	1.2	2.2			2.2
													AM.6	1.2	2.2	3.2		2.2
													AM.7	1.2	2.2	3.2		2.2
										X			APPLICATION INDEPENDENCE					
													AI.1		2.5	3.5		2.5
													AI.2		2.8		4.1,4.9	2.8
													AI.3		2.5		4.1	2.5
													AI.4		2.5			2.5
													AI.5		2.5	3.5	4.6	2.5
											X	X	AUGMENT-ABILITY					
													AG.1	1.6	2.6	3.6	(3.6),4.6	2.6
													AG.2	1.6	2.6	3.6	4.6,4.11	2.6
													AG.3	1.6	2.6	3.6	4.11	2.6
													AG.4	1.6	2.6			2.6
					X								AUTONOMY					
													AU.1	1.7	2.7	3.7	4.7,4.11	2.7
													AU.2	1.7	2.7			2.7
											X		COMMONALITY					
													CL.1	1.7	2.7			2.7
													CL.2	1.7	2.7			2.7
													CL.3	1.10				
				X							X		COMMUNICATIVE-NESS					
													CM.1	1.9	2.9			2.9
													CM.2	1.9	2.9			2.9
X													COMPLETENESS					
													CP.1	1.4	2.4	3.4	(3.4)	2.4
						X							CONCISENESS					
													CO.1				4.4	
X	X					X							CONSISTENCY					
													CS.1			3.8	(3.8)	
													CS.2	1.8	2.8	3.8	(3.8),4.9	2.8
					X								DISTRIBUTED-NESS					
													DI.1	1.1,1.8	2.1,2.8	3.1		2.1,2.8
										X			DOCUMENT ACCESSIBILITY					
													DA.1	1.11				
													DA.2	1.11				
													DA.3	1.11			4.6	
		X											EFFECTIVENESS					
													EF.1	1.3	2.3	3.3		
													EF.2		2.3	3.3	4.3,4.11	2.3
													EF.3		2.3	3.3	(3.3) 4.3, 4.9, 4.11	2.3
													EF.4		2.3		4.3,4.9, 4.11	2.3
											X		FUNCTIONAL OVERLAP					
													FO.1	1.13				

Table 3.3-1 METRIC WORKSHEET CORRELATION (continued)

QUALITY FACTORS														PHASES				
CORRECTNESS	RELIABILITY	EFFICIENCY	INTEGRITY	USABILITY	SURVIVABILITY	MAINTAINABILITY	VERIFIABILITY	FLEXIBILITY	PORTABILITY	REUSABILITY	INTEROPERABILITY	EXPANDABILITY	CRITERIA/ METRIC	REQUIREMENTS ANALYSIS	PRELIMINARY DESIGN	DETAILED DESIGN	IMPLEMENTATION	TEST & INTEGRATION
										X			FUNCTIONAL SCOPE					
													FS.1			3.1	4.6	
													FS.2	1.5			4.7	
													FS.3	1.5				
								X		X		X	GENERALITY					
													GE.1		2.6			2.6
													GE.2			3.6	4.6	
									X	X	X		INDEPENDENCE					
													ID.1			3.5	4.5,4.10	
													ID.2			3.5	(3.5),4.7 4.10	
					X	X	X	X	X	X	X	X	MODULARITY					
													MO.2		2.1	3.5	(3.5)4.5	
													MO.3	1.1	2.1	3.1		
				X									OPERABILITY					
													OP.1	1.9	2.9			2.9
					X								RECONFIGUR-ABILITY					
													RE.1	1.7,1.8	2.7,2.8			2.7,2.8
						X	X	X	X	X			SELF-DESCRIPTIVENESS					
													SD.1				4.8	
													SD.2				4.8,4.9	
													SD.3				4.8	
X	X					X	X	X		X		X	SIMPLICITY					
													SI.1	1.1	2.1,2.8	3.1	4.1	2.1,2.8
													SI.2				4.1	
													SI.3			3.1	4.1	
													SI.4	1.1		3.1	4.1,4.9	
X							X					X	SPECIFICITY					
													SP.1			3.1		
			X										SYSTEM ACCESSIBILITY					
													SA.1	1.12	2.12			2.12
													SA.2	1.12	2.12			2.12
										X			SYSTEM CLARITY					
													SC.1				4.1	
													SC.2				4.1	
													SC.3		2.1		4.1	2.1
													SC.4				4.1	
													SC.5				4.1	
											X		SYSTEM COMPATIBILITY					
													SY.1		2.11			2.11
													SY.2		2.11			2.11
													SY.3		2.11			
													SY.4		2.11			
													SY.5	1.11				
X													TRACEABILITY					
													TR.1	1.4	2.4	3.4		
				X									TRAINING					
													TN.1		2.9			2.9
			X	X								X	VIRTUALITY					
													VR.1	1.8	2.1, 2.8			2.1,2.8
				X		X	X						VISIBILITY					
													VS.1		2.10			2.10
													VS.2		2.10			2.10
													VS.3		2.10			2.10

( ) = Reapplication of Metric During Subsequent Phase

Definitions and interpretations of the individual measurements contained in the worksheets are found in Appendix C.

As shown in Figure 3.3-1, the worksheets may be applied several times during the development. For example, Metric Worksheet #3, which is applied for each module to the detailed design document during design, is also applied to the detailed design document after it has been updated to reflect the actual implementation. The worksheet does not have to be totally reapplied for each successive application. It should only involve updates to reflect the changes made to the system since the previous application of the worksheet. The successive applications of any worksheet should require considerably less effort than the original application.

Worksheet Number	Requirements Analysis	Preliminary Design	Detailed Design	Implementation	Test and Integration
1	Requirements Spec				
2		Preliminary Design Spec Users Manual (Draft)	Preliminary Design Spec Users Manual (Draft)		Test Results Users Manual (Final)
3			Detail Design Spec Test Plans & Procedures	Detail Design Spec Test Plans & Procedures	
4				Source Code Detail Design Spec (Updated)	

________ 1st Application
------ Reapplication

Figure 3.3-1 Application of the Metric Worksheets

3.4 TECHNIQUES FOR APPLYING METRICS

Section 1.5 identified organizational approaches for utilizing the quality metric concepts during software development. These approaches included both acquisition environments and internal development environments. The purpose of this section is to describe, at a lower level, how the metrics would be applied in either case.

The first technique for applying the metrics is by formal inspection. The formal inspection is performed by personnel of an organization independent of the development organization (the acquisition office, an independent quality assurance group, or an independent IV&V contractor). The metric worksheets are applied to delivered products at scheduled times and the results are formally reported.

The second technique is to utilize the worksheets during structured design and code walkthroughs held by the development team. A specific participant of the walkthrough can be designated to apply the worksheets and report any deficiencies during the walkthrough, or a quality assurance person can participate in the walkthroughs to take the measurements of the design or code.

The last technique is for the development team to utilize the worksheets as guidelines, self-evaluations or in a peer review mode to evaluate or enhance the quality of the products they produce.

SECTION 4
ASSESSING THE QUALITY OF THE SOFTWARE PRODUCT

4.1 INTRODUCTION

The benefits of applying the software quality metrics are realized when the information gained from their application is analyzed. The analyses that can be done are described in the subsequent paragraphs. There are three levels at which analyses can be performed. These levels are related to the level of detail to which the evaluating organization wishes to go in order to arrive at a quality assessment.

4.2 INSPECTOR'S ASSESSMENT

The first level at which an assessment can be made relies on the discipline and consistency introduced by the application of the worksheets. An inspector, using the worksheets, asks the same questions and takes the same counts for each module's source code or design document, etc. that is reviewed. Based on this consistent evaluation, a subjective comparison of products can be made.

1a. Document Inspector's Assessment
The last section in each worksheet is for the inspector to make comments on the quality observed while applying the worksheet. Comments should indicate an overall assessment as well as point out particular problem areas such as lack of comments, inefficiencies in implementation, or overly complex control flow.

1b. Compile Assessments for System Review
By compiling all of the inspector's assessments on the various documents and source code availabie at any time during the development, deficiencies can be identified.

4.3 SENSITIVITY ANALYSIS

The second level of detail utilizes experience gained through the application of metrics and the accumulation of historical information to take advantage of the quantitative nature of the metrics. The values of the measurements are used as indicators for evaluation of the progress toward the high quality goals or requirements.

At appropriate times during a large-scale development, the application of the worksheets allows calculation of the metrics. The correspondence of the worksheets to the metrics is shown in Appendix B. The results of these calculations is a matrix of measurements. The metrics that have been established to date are at two levels, system level and module level. The approach described is for the module level metrics however it is applicable to both levels.

A n by k matrix of measurements results from the application of the metrics to the existing products (e.g., at design, the products might include review material, design specifications, test plans, etc.) where there are k modules and n module level measurements applicable at this particular time.

$$M_d^m = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1k} \\ m21 & & \ddots & \\ \vdots & & & \vdots \\ m_{n1} & & & m_{nk} \end{bmatrix}$$

This matrix represents a profile of all the modules in the system with respect to a number of characteristics measured by the metrics. The analyses that can be performed are described in the following steps:

2a. <u>Assess Variation of Measurements</u>

Each row in the above matrix represents how each module in the system scored with respect to a particular metric. By summing all the values and calculating the average and standard deviation for that metric, each individual module's score can then be compared with the average. Those modules that score more than one standard deviation below the average should be reported for further examination. These calculations are illustrated below:

for metric i; Average Score = $A_i = \sum_{j=1}^{k} M_{ij}/k$

$$\text{Standard Deviation} = \sigma_i = \left(\sum_{j=1}^{k} (M_{ij}-A_i)^2/k\right)^{1/2}$$

Report Module j if $M_{ij} < A_i - \sigma_i$

2b. <u>Assess Low System Scores</u>

In examining a particular measure across all modules, consistently low scores may exist. It may be that a design or implementation technique used widely by the development team was the cause. This situation indicates the need for a new standard or stricter enforcement of existing standards to improve the overall development effort.

2c. <u>Assess Scores Against Thresholds</u>

As experience is gained with the metrics and data is accumulated, threshold values, or industry acceptable limits, may be established. The scores, for each module for a particular metric should be compared with the established threshold. A simple example is the percent of comments per line of source code. Certainly code which exhibits only one or two percent measurements for this metric would be identified for corrective action. It may be that ten percent is a minimum acceptable level. Another example is the complexity measure. A specific value of the complexity measure greater than some chosen value should be reported for corrective action.

Report Module j if $M_{ij} < T_i$

Where T_i = threshold value specified for metric i.

4.4 USE OF NORMALIZATION FUNCTION TO ASSESS QUALITY

The last level of assessing quality is using the normalization functions to predict the quality in quantitative terms. The normalization functions are utilized in the following manner.

At a particular time in the development process there is an associated matrix of coefficients which represent the results of linear multivariate regression analyses against empirical data (past software developments). These coefficients, when multiplied by the measurement matrix results in an evaluation (prediction) of the quality of the product based on the development to date. This coefficient matrix, shown below, has n columns for the coefficients of the various metrics and 13 rows for the 13 quality factors.

$$C_d^m = \begin{bmatrix} c_{1,1} & c_{1,2} \ldots c_{1,n} \\ \cdot \;\; \cdot & \\ \cdot \quad \cdot & \\ \cdot \qquad \cdot & \\ c_{13,1} & c_{13,n} \end{bmatrix}$$

To evaluate the current degree or level of a particular quality factor, i, for a module, j, the particular column in the measurement matrix is multiplied by the row in the coefficient matrix. The resultant value:

$$r_{i,j} = c_{i,1}\ m_{i,j} + c_{i,2}\ m_{2,j} \ldots + c_{i,n}\ m_{m,j}$$

is the current predicted rating of that module, j for the quality factor, i. This predicted rating is then compared to the previously established rating to determine if the quality is at least as sufficient as required. The coefficient matrix should be relatively sparse (many $C_{ij} = 0$). Only subsets of the entire set of metrics applicable at any one time relates to the criteria of any particular quality factor.

Multiplying the complete measurement matrix by the coefficient matrix results in a ratings matrix. This matrix contains the current predicted ratings of each module for each quality factor. Each module then can be compared with the preset rating for each quality factor.

$$CM = R_d^m = \begin{bmatrix} r_{1,1} & r_{1,2} \ldots r_{1,k} \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ r_{13,1} & r_{13,k} \end{bmatrix}$$

This represents the most formal approach to evaluating the quality of a product utilizing the software quality metrics. Because the coefficient matrix has been developed only for a limited sample in a particular environment, it is neither generally applicable nor has statistical confidence in its value been achieved.

To use the normalization functions that currently exist, the following steps should be performed.

3a. Apply Normalization Function

Table 4.4-1 contains the normalization functions that currently exist. If any of the quality factors identified in that table have been specified as a requirement, then the metrics identified in the table should be substituted into the equation and the predicted rating calculated. Normalization functions which include several metrics can be used if available, otherwise functions for individual metrics should be used. This predicted rating should be compared with the specified rating.

To illustrate the procedure, the normalization function that has been developed for the factor Flexibility will be used. The normalization function, applicable during the design phase, relates measures of modular implementation (MO.2) to the flexibility of the software. The predicted rating of flexibility is in terms of the average time to implement a change in specifications. The normalization function is shown in Figure 4.4-1. The measurements associated with the modular implementation metric are taken from design documents. The measurements involve identifying if input, output and processing functions are mixed in the same module, if application and machine-dependent functions are mixed in the same module and if processing is data volume limited. As an example, assume the measurements were applied during the design phase and a value of 0.65 was measured. Inserting this value in the normalization function results in a predicted rating for flexibility of .33 (.51 x .65) as identified by point A in Figure 4.4-1. If the

Acquisition Manager had specified a rating of 0.2, which is identified by point B, he has an indication that the software development is progressing well with respect to this desired quality.

An organization using this manual is encouraged to establish these functions in its specific environment by following the procedures described in (MCCA77), (MCCA80), or Volume 1 of this report.

Table 4.4-1 Normalization Functions

RELIABILITY (DESIGN)		
MULTIVARIATE FUNCTION	$.18\ M_{AM.1} + .19\ M_{SI.3}$	
INDIVIDUAL FUNCTIONS	$.34\ M_{AM.1}$ $.34\ M_{SI.3}$	AM.1 Error Tolerance/Control Checklist SI.3 Data and Control Flow Complexity Measure
RELIABILITY (IMPLEMENTATION)		
MULTIVARIATE FUNCTION	$.48\ M_{AM.1} + .14\ M_{SI.1}$	
INDIVIDUAL FUNCTIONS	$.57\ M_{AM.1}$ $.58\ M_{SI.1}$ $.53\ M_{SI.3}$ $.53\ M_{SI.4}$	AM.1 Error Tolerance/Control Checklist SI.1 Design Structure Measure SI.3 Data and Control Flow Complexity Measure SI.4 Coding Simplicity Measure
MAINTAINABILITY (DESIGN)		
INDIVIDUAL FUNCTIONS	$.57\ M_{SI.3}$ $.53\ M_{SI.1}$	SI.3 Data and Control Flow Complexity Measure SI.1 Design Structure Measure

Table 4.4-1 (Continued)

MAINTAINABILITY (IMPLEMENTATION)		
MULTIVARIATE FUNCTION	$-.2+.61\,M_{SI.3}+.14M_{MO.2}+.33_{SD.2}$	
INDIVIDUAL FUNCTIONS	$2.1\ M_{SI.3}$	SI.3 Data and Control Flow Complexity Measure
	$.71\ M_{SD.2}$	MO.2 Modular Implementation Measure
	$.6\ M_{SD.3}$	SD.2 Effectiveness of Comments Measure
	$.5\ M_{SI.1}$	SD.3 Descriptiveness of Language Measure
	$.4\ M_{SI.4}$	SI.1 Design Structure Measure
		SI.4 Coding Simplicity Measure
FLEXIBILITY (DESIGN)		
INDIVIDUAL FUNCTIONS	$.51\ M_{MO.2}$	MO.2 Modular Implementation Measure
	$.56\ M_{GE.2}$	GE.2 Implementation for Generality Checklist
FLEXIBILITY (IMPLEMENTATION)		
MULTIVARIATE FUNCTION	$.22M_{MO.2}+.44M_{GE.2}+.09M_{SD.3}$	
INDIVIDUAL FUNCTIONS	$.6\ M_{MO.2}$	MO.2 Modular Implementation Measure
	$.72M_{GE.2}$	GE.2 Implementation for Generality Checklist
	$.59\ M_{SD.2}$	SD.2 Effectiveness of Comments Measure
	$.56\ M_{SD.3}$	SD.3 Descriptiveness of Language Measure

Table 4.4-1 (Continued)

PORTABILITY (IMPLEMENTATION)		
MULTIVARIATE FUNCTION	$-1.7 + .19M_{SD.1} + .76M_{SD.2} + 2.5M_{SD.3} + .64M_{ID.2}$	
INDIVIDUAL FUNCTIONS	$1.07\ M_{SI.1}$	SD.1 Quantity of Comments
	$1.1\ M_{ID.1}$	SD.2 Effectiveness of Comments Measure
	$1.5\ M_{SD.2}$	SD.3 Descriptiveness of Language Measure
		ID.2 Machine Independence Measure
		SI.1 Design Structure Measure

REUSABILITY		
MULTIVARIATE FUNCTIONS	$.13 + .29\ M_{SI.1} + .08M_{SI.3}$	
	$.10 + .08M_{SD.1} + .19M_{SD.3} + .07M_{SI.3}$	
	$.11 + .04M_{FS.1} + .06M_{SD.1} + .16M_{SD.3} + .07M_{SI.3}$	
	$.11 + .03M_{FS.1} + .04M_{SC.4} + .06M_{SD.1} + .14M_{SD.3} + .06M_{SI.3}$	
INDIVIDUAL FUNCTIONS	$.22 + .12 * M_{FS.1}$	FS.1 Function Specificity
	$.05 + .28 * M_{GE.2}$	GE.2 Implementation for Generality Checklist
	$.14 + .17 * M_{ID.2}$	ID.2 Machine Independence Measure
	$.20 + .19 * M_{MO.2}$	MO.2 Modular Implementation Measure
	$.18 + .21 * M_{SC.1}$	SC.1 Interface Complexity
	$.22 + .14 * M_{SC.2}$	SC.2 Program Flow Complexity
	$.14 + .24 * M_{SC.4}$	SC.4 Communication Complexity
	$.23 + .16 * M_{SD.1}$	SD.1 Quantity of Comments
	$.01 + .36 * M_{SD.3}$	SD.3 Descriptiveness of Language Measure
	$.10 + .37 * M_{SI.1}$	
	$.26 + .13 * M_{SI.3}$	

Table 4.4-1 (Continued)

$-.14 + .56 * M_{SI.4}$	SI.1 Design Structure Measure SI.3 Data and Control Flow Complexity Measure SI.4 Coding Simplicity Measure

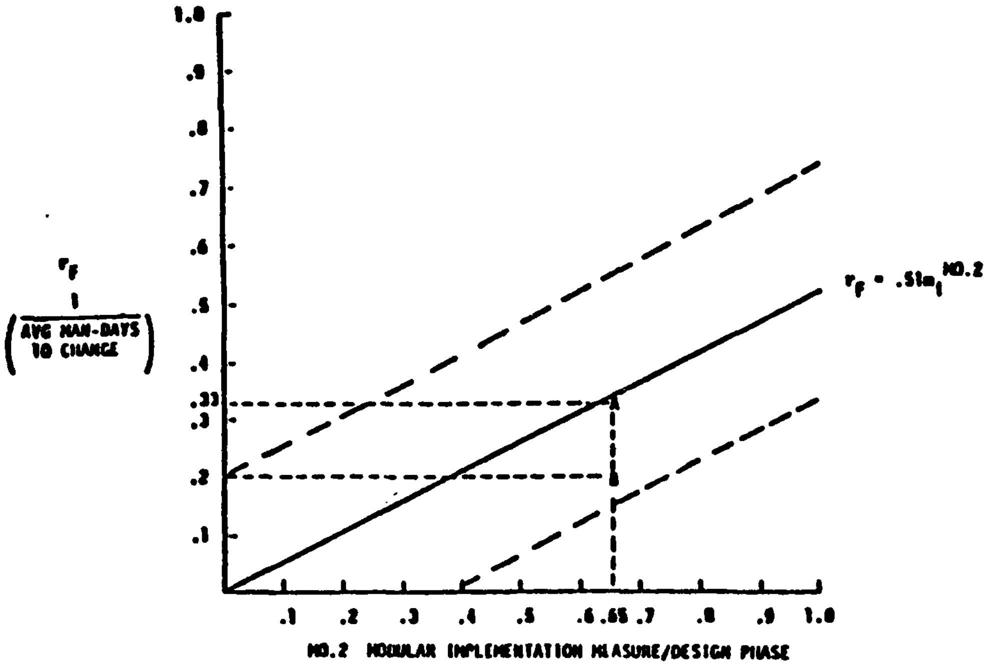
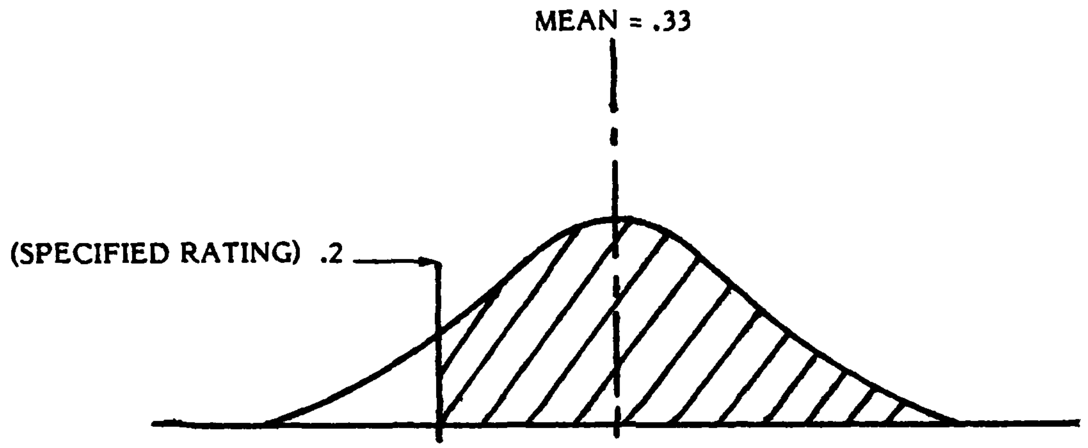


Figure 4.4.1 Normalization Function for Flexibility During Design

3b. Calculate Confidence in Quality Assessment

Using statistical techniques a level of confidence can be calculated. The calculation is based on the standard error of estimate for the normalization function and can be derived from a normal curve table found in most statistics texts. An example of the derivation process is shown in Figure 4.4-2 for the situation described above. Here it is shown that the Acquisition Manager has an 86 percent level of confidence that the flexibility of the system will be better than the specified rating.



MEAN = .33 (PREDICTED RATING)

STANDARD DEVIATION = .12 (STANDARD ERROR OF ESTIMATE)

LEVEL OF CONFIDENCE = $\Pr[X \geq .2]$ = .86 (SHADED AREA)

Figure 4.4-2 Determination of Level of Confidence

4.5 REPORTING ASSESSMENT RESULTS

Each of the preceding steps described in this section are easily automated. If the metrics are applied automatically then the metric data is available in machine readable form. If the worksheets are applied manually, then the data can be entered into a file, used to calculate the metric, and formatted into the measurement matrix format. The automation of the analyses involve simple matrix manipulations. The results of the analyses should be reported at various levels of detail. The formats of the reports are left to the discretion of the implementing organization. The content of the reports to the different managers is recommended in the following paragraphs.

1a. Report to the Acquisition Manager/Development Manager

The report content to the Acquisition Manager and the Development Manager should provide summary information about the progress of the development toward the quality goals identified at the beginning of the project.

For example if ratings were specified for several quality factors, the current predicted ratings should be reported.

QUALITY GOALS		PREDICTED RATING BASED ON DESIGN DOCUMENT
RELIABILITY	.9	.8
MAINTAINABILITY	.8	.95

If specific ratings were not identified but the important qualities were identified, a report might describe the percentage of modules that currently are judged to be below the average quality (as a result of the sensitivity analysis) or that are below a specified threshold value (as a result of the threshold analysis). These statistics provide a progress status report to the manager. Further progress status is indicated by reporting the quality growth of the system or of individual modules. The quality growth is depicted by reporting the scores achieved during the various phases of development. Ultimately the ratings should progressively score higher than those reported during the requirements phase. This progress is based on the identification of problems in the early phases which can then be corrected.

1b. Reports to Quality Assurance Manager

In addition to the summary quality progress reports described in 1a, the quality assurance manager and his staff will want detailed metric reports. These reports will provide all of the results of the Analyses described in 4.2, 4.3, and 4.4, and perhaps provide the measurement matrix itself for examinations. In addition to the detailed reports, the quality assurance manager should be provided with reports on the status of the application of the metrics themselves by the quality assurance staff. These status reports will provide information on the total number of modules and the number which inspectors have analyzed.

1c. Reports to the Development Team

The development team should be provided detailed information on an exception basis. This information is derived from the analyses. Examples of the information would be quality problems that have been identified, which characteristics or measurements of the software products are poor, and which modules have been identified as requiring rework. These exception reports should contain the details of why the assessment revealed them as potential problems. It is based on this information that corrective actions will be taken.

REFERENCES

(MCCA77) McCall, J., Richards, P., Walters, G., "Factors in Software Quality", RADC-TR-77-369, Nov 1977, 3 Vols (A049014) (A049015) & (A049055).

(MCCA80) McCall, J., Matsumoto, M., "Software Quality Metrics Enhancements", RADC-TR-80-109, April 1980.

(WEIN72) Weinberg, G., "The Psychology of Improved Programming Performance," DATAMATION, Nov 1972.

(CAVA78) Cavano, J., McCall, J., "A Framework for the Measurement of Software Quality," Proceedings of the ACM Software Quality Assurance Workshop, Nov 1978.

(DUVA76) Duvall, L.M., "Software Data Repository Study," RADC-TR-76-387, Dec 76, (A050636).

(POST82) Post, J.V., "The Role of Measurements in the Software Development Process", Proceeding COMSAC-82 (IEEE Computer Society Sixth International Computer Software and Applications Conference) Chicago, November 1982.

Additional references are contained in Appendix A of Volume III.

APPENDIX A
METRIC WORKSHEETS

Appendix A contains the metric worksheets which are used to gather metric data during the software development phases. There are four worksheets, organized by applicable phase:

Worksheet 1 - Requirements Analysis
Worksheet 2 - Preliminary Design
Worksheet 3 - Detailed Design
Worksheet 4 - Source Code

A summary of the worksheets is shown on the next page. Each worksheet is divided into sections of related questions to ease the data gathering task. The applicable metric element is referenced by acronym at the end of each worksheet question. Appendix B, Metric Tables, lists the formula to be used in calculating values for metrics and metric elements.

The contents of this appendix are based on the results of this contract, "Quality Metrics for Distributed Systems", F30602-80-C-0330 and the results of contract F30602-80-C-0265, "Software Interoperability and Reusability". This appendix includes a refinement and reorganization of worksheet information initially defined in RADC-TR-77-369 and RADC-TR-80-109.

METRIC WORKSHEETS SUMMARY

1 REQUIREMENTS ANALYSIS/SYSTEM LEVEL

1.1 Structure (CO, RL, SV, MA, VE, FX, PO, RU, IP, EX)
SI.1, DI.1 SI.4, MO.3

1.2 Tolerance (RL, SV)
AY.1, AM.2, AM.3, AM.4, AM.5, AM.6, AM.7

1.3 Performance (EF)
EF.1

1.4 Completeness (CO)
CP.1, TR.1

1.5 Functional Scope (RU)
FS.2, FS.3

1.6 *Changeability* (IP, EX)
AG.1, AG.2, AG.3, AG.4

1.7 System Interfaces (SV, IP)
CL.1, CL.2, AU.1, AU.2, RE.1

1.8 Data Base (CO, RL, IG, US, SV, MA, EX)
RE.1, DI.1, VR.1, CS.2

1.9 Human Interface (US, IP)
OP.1, CM.1, CM.2

1.10 Common Vocabulary (IP)
CL.3

1.11 Documentation (IP, RU)
DA.1, DA.2, DA.3, SY.5

1.12 Security (IG)
SA.1, SA.2

1.13 Functional Overlap (IP)
FO.1

1.14 Inspector's Comments

2 PRELIMINARY DESIGN/SYSTEM LEVEL

2.1 Structure (CO, RL, IG, US, SV, MA, VE, FX, PO, RU, IP, EX)
SI.1, DI.1, VR.1, MO.2, MO.3, SC.3

2.2 Tolerance (RL, SV)
AY.1, AM.1, AM.4, AM.5, AM.6, AM.7

2.3 Optimization (EF)
EF.1, EF.2, EF.3, EF.4

2.4 Completeness (CO)
TR.1, CP.1

2.5 References (RU)
AI.1, AI.3, AI.4, AI.5

2.6 Changeability (FX, RU, IP, EX)
AG.1, AG.2, AG.3, AG.4, GE.1,

2.7 System Interfaces (SV, IP)
CL.1, CL.2, AU.1, AU.2, RE.1

2.8 Data Base (CO, RL, IG, US, SV, MA, VE, FX, RU, EX)
AI.2, SI.1, RE.1, DI.1, VR.1, CS.2

2.9 Human Interface (US, IP)
OP.1, TN.1, CM.1, CM.2

2.10 Testing (US, MA, VE)
VS.1, VS.2, VS.3

2.11 System Compatibility (IP)
SY.1, SY.2, SY.3, SY.4

2.12 Security (IG)
SA.1, SA.2

2.13 Inspector's Comments

3 DETAILED DESIGN/MODULE LEVEL

3.1 Structure (CO, RL, SV, MA, VE, FX, PO, RU, IP, EX)
SI.1, SI.3, SI.4, DI.1, SP.1, MO.3, FS.1

3.2 Tolerance (RL, SV)
AY.1, AM.1, AM.2, AM.3, AM.6, AM.7

3.3 Optimization (EF)
EF.1, EF.2, EF.3

3.4 Completeness (CO)
CP.1, TR.1

3.5 References (SV, MA, VE, FX, PO, RU, IP, EX)
AI.1, AI.5, ID.1, ID.2, MO.2

3.6 Changeability (FX, RU, IP, EX)
AG.1, AG.2, AG.3, GE.2

3.7 System Interfaces (SV)
AU.1

3.8 Consistency (CO, RL, MA)
CS.1, CS.2

3.9 Functional Categorization

3.10 Inspector's Comments

4 SOURCE CODE/MODULE LEVEL

4.1 Structure (CO, RL, SV, MA, VE, FX, PO, RU, EX)
AI.2, AI.3, SI.1, SI.2, SI.3, SI.4, SC.1, SC.2, SC.3, SC.4, SC.5,

4.2 Tolerance (RL, SV)
AM.1, AM.3

4.3 Optimization (EF)
EF.2, EF.3, EF.4

4.4 Conciseness (MA)
CO.1

4.5 References (SV, MA, VE, FX, PO, RU, IP, EX)
MO.2, ID.1

4.6 Changeability (FX, RU, EX, IP)
AI.5, AG.1, AG.2, GE.2, DA.3, FS.1

4.7 Input/Output (RL, SV, PO, RU, IP)
ID.2, AM.2, AU.1, FS.2

4.8 Self-Descriptiveness (MA, VE, FX, PO, RU)
SD.1, SD.2, SD.3

4.9 Data (CO, RL, EF, MA, VE, FX, RU, EX)
AI.2, SI.4, EF.3, EF.4, CS.2, SD.2

4.10 Independence (PO, RU, IP)
ID.1, ID.2

4.11 Dynamic Measurement (RL, EF, SV, FX, EX)
AY.1, AG.2, AG.3, EF.2, EF.3, EF.4, AU.1

4.12 Inspector's Comments

METRIC WORKSHEET 1	SYSTEM:	DATE:
REQUIREMENTS ANALYSIS/SYSTEM LEVEL	NAME:	INSPECTOR:

1.1 STRUCTURE (RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, REUSABILIT EXPANDABILITY, SURVIVABILITY, PORTABILITY, INTEROPERABILITY, CORRECTNESS)

1.	Is an organization of the system/network provided which identifies all software functions and functional interfaces in the system? DI.1(1)	Y	N
2.	Number of major functions. SI.1(2)		
3.	Are there no duplicate functions? SI.1(2)	Y	N
4.	Is there a definitive statement of the requirements for the distibution of information within the data base? DI.1(3)	Y	N
5.	Is there an organization of the data base provided which identifies the types of system-level information and the information flow within the system? DI.1(2)	Y	N
6.	Is there a definitive statement of requirements for code to be written according to a programming standard? SI.4(13)	Y	N
7.	Is there a definitive statement of requirements for processes, functions, and modules to have loose coupling? MO.3(1)	Y	N
8.	Is there a definitive statement of requirements for processes, functions, and modules to have high cohesion? MO.3(2)	Y	N

1.2 TOLERANCE (RELIABILITY, SURVIVABILITY)

1.	Has an error analysis been performed and budgeted to functions? AY.1(1)	Y	N
2.	Are there definitive statements of the accuracy requirements for inputs, outputs, processing, and constants? AY.1(2)	Y	N
3.	Are there definitive statements of the error tolerance of input data? AM.2(1)	Y	N
4.	Are there definitive statements of the requirements for recovery from computational failures? AM.3(1)	Y	N
5.	Is there a definitive statement of the requirement for recovery from hardware faults? AM.4(1)	Y	N
6.	Is there a definitive statement of the requirements for recovery from device errors? AM.5(1)	Y	N
7.	Are there definitive statements of the requirements for recovery from communication errors? AM.6(1)	Y	N

METRIC WORKSHEET 1	SYSTEM:	DATE:
REQUIREMENTS ANALYSIS/SYSTEM LEVEL	NAME:	INSPECTOR:

8.	Are there definitive statements of the requirements for system recovery from node or communication failures? AM.7(1)	Y N
1.3	**PERFORMANCE (EFFICIENCY)**	
1.	Have performance requirements and limitations (flow time for process, including execution and communication; storage) been specified for the functions to be performed? EF.1(1)	Y N
1.4	**COMPLETENESS (CORRECTNESS)**	
1.	Is there a matrix relating itemized requirements to major functions which implement those requirements? TR.1(1)	Y N
2.	Number of major functions identified (equivalent to CPCI). CP.1	
3.	Are requirements itemized so that the various functions to be performed, their inputs and outputs, are clearly delineated? CP.1(1)	Y N
4.	Number of major data references. CP.1(2)	
5.	How many of these data references are not defined? CP.1(2)	
6.	How many defined functions are not used? CP.1(3)	
7.	How many referenced functions are not defined? CP.1(4)	
8.	How many data references are not used? CP.1(2)	
9.	How many referenced data references are not defined? CP.1(6)	
10.	Is the flow of processing and all decision points in that flow described. CP.1(5)	Y N
11.	How many problem reports related to the requirements have been recorded? CP.1(7)	
12.	How many of those problem reports have been closed (resolved)? CP.1(7)	
1.5	**FUNCTIONAL SCOPE (REUSABILITY)**	
1.	Is the function constructed in such a way to encourage its use elsewhere either in part or in total? FS.2(1)	Y N
2.	Are the input quantities well defined? FS.2(2)	Y N
3.	Are the output well defined and easy to interpret? FS.2(4)	Y N
4.	Do the functions performed satisfy one of the specified requirements? FS.2(5)	Y N
5.	Number of function requirements satisfied by the reusable software? FS.3(1)	
6.	Total number of requirements? FS.3(1)	

METRIC WORKSHEET 1	SYSTEM:	DATE:
REQUIREMENTS ANALYSIS/SYSTEM LEVEL	NAME:	INSPECTOR:

1.6 CHANGEABILITY (INTEROPERABILITY, EXPANDABILITY)

1.	Is there a definitive statement of requirements for spare storage capacity (memory and auxiliary storage)? AG.1(2,3)	Y	N
2.	Is there a definitive statement of requirements for spare processing capacity? AG.2(3)	Y	N
3.	Is there a definitive statement of requirements for spare I/O and communication channel capacity? AG.3(1,2)	Y	N
4.	Is there a definitive statement of requirements for interface compatibility among all the processors, communication links, memory devices, and peripherals? AG.4(1)	Y	N
5.	Is there a specific requirement for providing performance/price information for enhancement trades? AG.4(2)	Y	N
6.	Do specifications identify new technology tradeoff areas for software? AG.4(3)	Y	N
7.	Do software specifications include requirements for the criteria of the quality factor expandability? AG.4(4)	Y	N

1.7 SYSTEM INTERFACES (INTEROPERABILITY, SURVIVABILITY)

1.	Is there a definitive statement of the requirements for communication with other systems? CL.1(1)	Y	N
2.	Are there specific requirements for network process control? CL.1(5)	Y	N
3.	Are there specific requirements for user session control? CL.1(6)	Y	N
4.	Are there specific requirements for a communication routing strategy? CL.1(7)	Y	N
5.	Is there a definitive statement of the requirements for standard data representations for communication with other systems? CL.2(1)	Y	N
6.	Are processes and functions separated as logical "wholes" to minimize interface complexity? AU.1(1)	Y	N
7.	Are there specific requirements for each CPU/system to have a separate power source? AU.2(1)	Y	N
8.	Are there specific requirements for each software scheduling unit to test its own operation, communication links, memories, and peripherals? AU.2(3)	Y	N
9.	Are there specific requirements for the software system to include a word processing capability? AU.2(3)	Y	N
10.	Are there specific requirements for network communication capabilities in the event of failure of a node or communication link? RE.1(1)	Y	N

METRIC WORKSHEET 1	SYSTEM:	DATE:
REQUIREMENTS ANALYSIS/SYSTEM LEVEL	NAME:	INSPECTOR:

11.	Are there specific requirements for a node to rejoin the network when it has been recovered? RE.1(4)	Y	N
12.	Is there a definitive statement of the operating procedures to be used with this system? CL.1(15)	Y	N
13.	Is there a low dependency on handshaking time between systems? CL.1(11)	Y	N
14.	How many systems must respond correctly to successfully complete handshaking? CL.1(10)		
15.	Are there no timing dependencies on the system communication response time that effect system performance requirements? CL.1(12)	Y	N
16.	Are there no timing dependencies on the freshness of data that effect system performance requirements? CL.1(14)	Y	N

1.8 DATA BASE (SURVIVABILITY, USABILITY, INTEGRITY, EXPANDABILITY, CORRECTNES RELIABILITY, MAINTAINABILITY)

1.	Is there a definitive statement of the requirements for maintaining data base integrity under anomalous conditions? RE.1(2)	Y	N
2.	Are there specific requirements for file/library accessibility from each node? DI.1(4)	Y	N
3.	Are there specific requirements for a virtual storage structure? VR.1(1)	Y	N
4.	Is there a definitive statement of the requirements for establishing and verifying data base consistency and concurrency at each node which hosts a data base partition? CS.2(4)	Y	N

1.9 HUMAN INTERFACE (USABILITY, INTEROPERABILITY)

1.	Are all steps in the operation described (operations concept)? OP.1(1)	Y	N
2.	Are all error conditions to be reported to operator/user identified and the responses described? OP.1(2)	Y	N
3.	Is there a statement of the requirement for the capability to interrupt operation, obtain operational status, save, modify, and continue processing? OP.1(3)	Y	N
4.	Is there a statement of the requirement for the capability to obtain network resource status? OP.1(9)	Y	N
5.	Is there a definitive statement of requirements for optional input media? CM.1(6)	Y	N
6.	Is there a definitive statement of requirements for optional output media? CM.2(7)	Y	N
7.	Is there a definitive statement of requirements for selective output control? CM.2(1)	Y	N
8.	Is there a definitive statement of requirements for selection of different nodes for different types of processing or for different types of information retrieval? OP.1(10)	Y	N

METRIC WORKSHEET 1	SYSTEM:	DATE:
REQUIREMENTS ANALYSIS/SYSTEM LEVEL	NAME:	INSPECTOR:

9.	Is there a definitive statement of requirements for establishing standard user interfaces for network information and data access? CM.2(8)	Y	N
1.10	**COMMON VOCABULARY (INTEROPERABILITY)**		
1.	Do both projects use the same technical vocabulary with identical meanings? CL.3(1)	Y	N
1.11	**DOCUMENTATION (REUSABILITY, INTEROPERABILITY)**		
1.	Is there no access control to the software document? DA.1(1)	Y	N
2.	Are the documents clearly and simply written? DA.2(1)	Y	N
3.	Do the documents contain software flow charts with adequate information and explanation? DA.2(2)	Y	N
4.	Do the documents have hierarchical structured table of contents? DA.2(3)	Y	N
5.	Do the documents have index system? DA.2(4)	Y	N
6.	Do the documents have separate volumes based on function? DA.2(5)	Y	N
7.	Do the documents have functional range of the system? DA.2(6)	Y	N
8.	Do the documents describe the functions performed? DA.2(7)	Y	N
9.	Do the documents describe the algorithm used and limitations? DA.2(8)	Y	N
10.	Do the documents describe the relationship between functions? DA.2(9)	Y	N
11.	Do the documents contain the software program listing? DA.2(10)	Y	N
12.	Do the programs have selective computation/output options? DA.3(1)	Y	N
13.	Are the functions performed generally associated with request application? DA.3(3)	Y	N
14.	Is the other system documentation available in a form that is up-to-date, complete and clearly organized? SY.5(1)	Y	N
1.12	**SECURITY (INTEGRITY)**		
1.	Is there a definitive statement of the requirements for user input/output access controls? SA.1(1)	Y	N
2.	Is there a definitive statement of the requirements for data base access controls? SA.1(2)	Y	N
3.	Is there a definitive statement of the requirements for memory protection across task? SA.1(3)	Y	N

METRIC WORKSHEET 1	SYSTEM:	DATE:
REQUIREMENTS ANALYSIS/SYSTEM LEVEL	NAME:	INSPECTOR:

4. Is there a definitive statement of the requirements for recording and reporting access to system? SA.2(1)	Y	N
5. Is there a definitive statement of the requirements for immediate indication of access violation? SA.2(2)	Y	N
6. Is there a definitive statement of the requirements for network access controls? SA.1(4)	Y	N

1.13 FUNCTIONAL OVERLAP (INTEROPERABILITY)

1. How many functions are duplicated in the systems that are to interoperate? FO.1(1)	
2. How many of these duplicated functions will be deleted in one or the other system? FO.1(2)	
3. How many of these duplicated function pairs will require to be synchronized? FO.1(3)	
4. How many of these duplicated function pairs will require redundancy management logic to combine them? FO.1(4)	

1.14 INSPECTOR'S COMMENTS

Make any general or specific comments that relate to the quality observed while applying th checklist.

METRIC WORKSHEET 2	SYSTEM:	DATE:
DESIGN/SYSTEM LEVEL	NAME:	INSPECTOR:

2.1 STRUCTURE (RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, REUSABILIT EXPANDABILITY, SURVIVABILITY, PORTABILITY, INTEROPERABILITY, INTEGRITY, USABILITY, CORRECTNESS)

1.	Is an organization of the system provided which identifies all functions and functional interfaces in the system? DI.1(1)	Y	N
2.	Is a hierarchy of system identifying all modules in the system provided? SI.1(1)	Y	N
3.	Are there no duplicate functions or modules? SI.1(2)	Y	N
4.	Is an organization of the data base provided which identifies all functional groupings of data and data flow within the system? DI.1(2)	Y	N
5.	Are there provisions for selecting alternate processing capabilities? DI.1(5)	Y	N
6.	Are critical system functions distributed over redundant elements or nodes? DI.1(6)	Y	N
7.	Does the distribution of control functions ensure network operation/integrity under anomalous conditions? DI.1(7)	Y	N
8.	Are logical structure and function separated in the design? DI.1(8)	Y	N
9.	Are physical structure and function separated in the design? DI.1(9)	Y	N
10.	Number of nodes that can be removed and still have each node able to communicate with each remaining node. DI.1(10)		
11.	Do processes and functions have loose coupling? MO.3(1)	Y	N
12.	What is the cohesion value of processes and functions? MO.3(2)		
13.	Can each user utilize the system as though it were dedicated to that user? VR.1(4)	Y	N
14.	Is the user presented with a complete logical system without regard to physical topology? VR.1(5)	Y	N
15.	Do module descriptions include identification of module interfaces? SI.1(9)	Y	N
16.	Has a programming standard been developed? SI.1(8)	Y	N
17.	Number of modules with mixed input/output and computational functions? SC.3(1)		
18.	Is the common function not distributed in different modules? SC.3(4)	Y	N
19.	Does the module not perform many (related but different) functions? SC.3(5)	Y	N
20.	Number of modules which do not perform single function. MO.2(8)		
21.	Are the modules hierarchically constructed? MO.2(1)	Y	N

2.2 TOLERANCE (RELIABILITY, SURVIVABILITY)

1.	Have accuracy requirements been budgeted to functions? AY.1(6)	Y	N

METRIC WORKSHEET 2	SYSTEM:	DATE:
DESIGN/SYSTEM LEVEL	NAME:	INSPECTOR:

2.	Have math library routines to be used been checked for sufficiency with regards to accuracy requirements? AY.1(3)	Y	N
3.	Is concurrent processing centrally controlled? AM.1(1)	Y	N
4.	Is parallel processing centrally controlled? AM.1(4)	Y	N
5.	How many error conditions are reported by the system? AM.1(2)		
6.	How many of those errors are automatically fixed or bypassed and processing continues? AM.1(2)		
7.	How many, require operator intervention? AM.1(2)		
8.	Are there provisions for recovery from hardware faults? AM.4(2)	Y	N
9.	Are there provisions for recovery from device errors? AM.5(2)	Y	N
10.	Are there provisions for recovery from communication errors? AM.6(2)	Y	N
11.	Are there provisions for system recovery from node or communication failures? AM.7(2)	Y	N

2.3 OPTIMIZATION (EFFICIENCY)

1.	Have storage requirements and limitations been allocated to functions? EF.4(1)	Y	N
2.	Are virtual storage facilities used? EF.4(2)	Y	N
3.	Is dynamic memory management used? EF.4(5)	Y	N
4.	Is a performance optimizing compiler used? EF.4(7)	Y	N
5.	Have Data Base or files been organized for efficient processing? EF.3(1,5)	Y	N
6.	Are data base files/libraries stored at only one node? EF.4(8)	Y	N
7.	Is data packing used? EF.2(5)	Y	N
8.	Number of overlays EF.2(4)		
9.	Overlay efficiency - memory allocation EF.2(4) max overlay size		
	min overlay size		
10.	Has program been segmented for efficient storage? EF.4(4)	Y	N
11.	Have performance requirements and limitations been allocated to functions? EF.1(1)	Y	N

2.4 COMPLETENESS (CORRECTNESS)

1.	Is there a matrix relating system level requirements to functions which implement those requirements? TR.1(1)	Y	N

METRIC WORKSHEET 2	SYSTEM:	DATE:
DESIGN/SYSTEM LEVEL	NAME:	INSPECTOR:

2. How many major functions (CPCI's) are identified? CP.1
3. Are requirements itemized in such a way that the functions to be performed, their inputs and outputs are clearly delineated? CP.1(1) Y N
4. How many functions identified are not defined? CP.1(4)
5. How many defined functions are not used? CP.1(3)
6. How many interfaces between functions are not defined? CP.1(6)
7. Number of total problem reports recorded? CP.1(7)
8. Number of those reports that have not been closed (resolved)? CP.1(7)
9. Profile of problem reports: (number of following types)

a. Computational	h. Routine/System Interface	p. Recurrent errors
b. Logic		q. Documentation
c. Input/output	i. Tape Processing	r. Requirement compliance
d. Data handling	j. User interface	s. Operator
e. OS/System Support	k. Data base interface	t. Questions
f. Configuration	l. User requested changes	u. Hardware
g. Routine/Routine Interface		v. Network protocol
	m. Preset data	w. Communication routing
	n. Global variable definition	

2.5 REFERENCES (REUSABILITY)

1. Number of modules with database system reference. AI.1(1)
2. Number of modules with computer architecture reference. AI.3(1)
3. Number of modules are not in standard computer architecture. AI.3(2)
4. Number of modules used microcode instruction statements. AI.4(1)
5. Number of modules used the table driven algorithm. AI.5(2)

2.6 CHANGEABILITY (FLEXIBILITY, REUSABILITY, EXPANDABILITY, INTEROPERABILITY)

1. Percent of memory capacity uncommitted. AG.1(2)
2. Percent of auxiliary storage capacity uncommitted. AG.1(3)
3. Percent of speed capacity uncommitted. AG.2(3)

METRIC WORKSHEET 2	SYSTEM:	DATE:
DESIGN/SYSTEM LEVEL	NAME:	INSPECTOR:

4.	Spare I/O channel capacity. AG.3(1)	
5.	Spare communication channel capacity. AG.3(2)	
6.	Are processors, communication links, memory devices, and peripherals compatible (of a common vendor or model)? AG.4(1)	Y N
7.	Does documentation reveal performance/price of software/system for enhancement trades? AG.4(2)	Y N
8.	Do specifications identify new technology tradeoff areas for software? AG.4(3)	Y N
9.	Do software specifications include requirements for the criteria of the quality factor expandability. AG.4(4)	Y N
10.	Based on hierarchy or a call/called matrix, how many modules are called by more than one module? GE.1(1)	
11.	Number of modules. GE.1(1)	

?.7 SYSTEM INTERFACES (INTEROPERABILITY, SURVIVABILITY)

1.	How many nodes will this network/system interface with? CL.1(1)	
2.	Have protocol standards been established for network process control? CL.1(2)	Y N
3.	Have protocol standards been established for user session control? CL.1(8)	Y N
4.	Have protocol standards been established for communication routing? CL.1(9)	Y N
5.	Are they being complied with? CL.1(2)	Y N
6.	Number of modules used for input to other systems? CL.1(3)	
7.	Number of modules used for output to other systems? CL.1(4)	
8.	Has a standard data representation been established or translation standards between representations been established? Are they being compiled with? CL.2(2)	Y N
9.	Number of modules used to perform translations? CL.2(3)	
10.	Is configuration of communication links such that failure of one node/link will not disable communication among other nodes? RE.1(1)	Y N
11.	Can node rejoin the network when it has been recovered? RE.1(4)	Y N
12.	Is data replicated at two or more distinct nodes? RE.1(5)	Y N
13.	Are processes and functions separated as logical "wholes" to minimize interface complexity? AU.1(1)	Y N
14.	Estimated number of lines of interface code. AU.1(2)	
15.	Estimated number of interface modules. AU.1(3)	

METRIC WORKSHEET 2	SYSTEM:	DATE:
DESIGN/SYSTEM LEVEL	NAME:	INSPECTOR:

16.	Estimated time engaged in communication. AU.1(4)	
17.	Does each CPU/system have a separate power source? AU.2(1)	Y N
18.	Does each scheduling unit test its own operation, communication links, memories, and peripherals? AU.2(2)	Y N
19.	Does the software system include a word-processing capability? AU.2(3)	Y N
20.	How many other systems will this system interface with? CL.1(13)	

2.8 DATA BASE (RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, REUSABILIT EXPANDABILITY, USABILITY, INTEGRITY, SURVIVABILITY, CORRECTNESS)

1.	Number of unique data items in data base SI.1(6)	
2.	Number of preset data items SI.1(6)	
3.	Number of major segments (files) in data base SI.1(7)	
4.	Is the data base structured so that at least one copy of a file/library resides at a node which is accessible to all other nodes? DI.1(4)	Y N
5.	Is the data base structured so that users need not care about changes in the actual storage structure of data? VR.1(2)	Y N
6.	Are there provisions for maintaining data base integrity under anomalous conditions? RE.1(3)	Y N
7.	Can users manipulate data as if it were not replicated elsewhere in the system? VR.1(3)	Y N
8.	Have procedures been established for verifying data base consistency and concurrency at each node which hosts a data base partition? CS.2(5)	Y N
9.	Are all data centrally controlled and symbolically defined and referenced? AI.2(3)	Y N

2.9 HUMAN INTERFACE (USABILITY, INTEROPERABILITY)

1.	Are all steps in operation described including alternative flows? OP.1(1)	Y N
2.	Number of operator actions? OP.1(4)	
3.	Estimated or Actual time to perform? OP.1(4)	
4.	Budgeted time for complete job? OP.1(4)	
5.	Are job set up and tear down procedures described? OP.1(5)	Y N
6.	Is a hard copy of operator interactions to be maintained? OP.1(6)	Y N
7.	Number of operator messages and responses? OP.1(7)	
8.	Number of different formats? OP.1(7)	
9.	Are all error conditions and responses appropriately described? OP.1(2)	Y N
10.	Are all access violations and responses appropriately described? OP.1(8)	Y N

METRIC WORKSHEET 2	SYSTEM:	DATE:
DESIGN/SYSTEM LEVEL	NAME:	INSPECTOR:

11.	Does the capability exist for the operator to interrupt, obtain status, save, modify, and continue processing? OP.1(3)	Y N
12.	Does the capability exist for the operator to obtain network resource status. OP.1(9)	Y N
13.	Can different nodes be selected for different types of processing or for different types of information retrieval? OP.1(10)	Y N
14.	Are lesson plans/training materials for operators, end users, and maintainers provided? TN.1(1)	Y N
15.	Are realistic, simulated exercises provided? TN.1(2)	Y N
16.	Are help and diagnostic information available? TN.1(3)	Y N
17.	Number of different input record formats CM.1(2)	
18.	Number of input values CM.1(3)	
19.	Number of default values CM.1(1)	
20.	Total number of parameters CM.1(1)	
21.	Number of self-identifying input values CM.1(3)	
22.	Can input be verified by user prior to execution? CM.1(4)	Y N
23.	Is input terminated by explicitly defined by logical end of input? CM.1(5)	Y N
24.	Can input be specified from different media? CM.1(6)	Y N
25.	Are there selective output controls? CM.2(1)	Y N
26.	Do outputs have unique descriptive user oriented labels? CM.2(2)	Y N
27.	Do outputs have user oriented units? CM.2(3)	Y N
28.	Number of different output formats? CM.2(4)	
29.	Are logical groups of output separated for user examination? CM.2(5)	Y N
30.	Are relationships between error messages and outputs unambiguous? CM.2(6)	Y N
31.	Are there provisions for directing output to different media? CM.2(7)	Y N
32.	Are there standards governing the user interface for network information and data access? CM.2(8)	Y N
33.	Are the standards being complied with? CM.2(8)	Y N
34.	Are there selectable levels of aid and guidance for users of different degrees of expertise? TN.1(4)	Y N

2.10 TESTING (USABILITY, MAINTAINABILITY, VERIFIABILITY)—APPLY TO TEST PLAN, PROCEDURE RESULTS

1.	Number of paths? VS.1(1)	
2.	Number of paths to be tested? VS.1(1)	

METRIC WORKSHEET 2	SYSTEM:	DATE:
DESIGN/SYSTEM LEVEL	NAME:	INSPECTOR:

3.	Number of input parameters? VS.1(2)	
4.	Number of input parameters to be tested? VS.1(2)	
5.	Number of interfaces? VS.2(1)	
6.	Number of interfaces to be tested? VS.2(1)	
7.	Number of itemized performance requirements? VS.2(2)	
8.	Number of performance requirements to be verified? VS.2(2)	
9.	Number of modules? VS.3(1)	
10.	Number of modules to be exercised. VS.3(1)	
11.	Are test inputs and outputs provided in summary form? VS.3(2)	Y N

2.11 SYSTEM COMPATIBILITY (INTEROPERABILITY)

1.	Same I/O transmission rates in both systems? SY.1(1)	Y N
2.	Same communication protocol in both systems? SY.1(2)	Y N
3.	Same message content in both systems? SY.1(3)	Y N
4.	Same message structure and sequence in both systems? SY.1(4)	Y N
5.	Is data in both systems in the same format (ASCII, EBCDIC,...)? SY.2(1)	Y N
6.	Same data base structure in both systems? SY.2(2)	Y N
7.	Same data base access techniques in both systems? SY.2(3)	Y N
8.	Same source language in both systems? SY.4(1)	Y N
9.	Same operating system in both systems? SY.4(2)	Y N
10.	Same support software in both systems? SY.4(3)	Y N
11.	Same word length in both systems? SY.3(1)	Y N
12.	Same interrupt structure in both systems? SY.3(2)	Y N
13.	Same instruction set in both systems? SY.3(3)	Y N

2.12 SECURITY (INTEGRITY)

1.	Are user Input/Output access controls provided? SA.1(1)	Y N
2.	Are Data Base access controls provided? SA.1(2)	Y N
3.	Is memory protection across tasks provided? SA.1(3)	Y N
4.	Are there provisions for recording and reporting access to system? SA.2(1)	Y N
5.	Are network access controls provided? SA.1(4)	Y N
6.	Are there provisions for immediate indication of access violation? SA.2(2)	Y N

METRIC WORKSHEET 2	SYSTEM:	DATE:
DESIGN/SYSTEM LEVEL	NAME:	INSPECTOR:

2.13 INSPECTOR'S COMMENTS

Make any general or specific comments about the quality observed while applying this checklist.

METRIC WORKSHEET 3	SYSTEM NAME:	DATE:
DESIGN/MODULE LEVEL	MODULE NAME:	INSPECTOR:

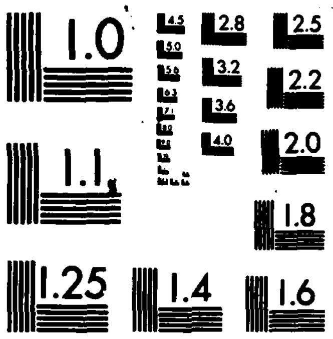
3.1 STRUCTURE (RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, REUSABILIT EXPANDABILITY, CORRECTNESS, PORTABILITY, INTEROPERABILITY, SURVIVABILITY)

1.	Is an organization of the system provided which identifies all modules and module interfaces? DI.1(1)	Y	N
2.	Is an organization of the data base provided which identifies all data base modules and module interfaces? DI.1(2)	Y	N
3.	How many decision points are there? SI.3(1)		
4.	How many subdecision points are there? SI.3(1)		
5.	How many conditional branches are there? SI.3(1)		
6.	How many unconditional branches are there? SI.3(1)		
7.	Is the module dependent on the source of the input or the destination of the output? SI.1(3)	Y	N
8.	Is the module dependent on knowledge of prior processing SI.1(3)	Y	N
9.	Number of entrances into modules SI.1(5)		
10.	Number of exits from module SI.1(5)		
11.	Does the module description include input, output, processing, and limitations? SI.1(4)	Y	N
12.	Is code written according to a programming standard? SI.4(13)	Y	N
13.	Are macros and subroutines used to avoid repeated and redundant code? SI.4(14)	Y	N
14.	Number of input parameters. SP.1(1)		
15.	Number of output values used. SP.1(2)		
16.	Number of output parameters. SP.1(2)		
17.	Can the same function not be accomplished by multiple variant forms? SP.1(3)	Y	N
18.	Does each function and module have loose coupling? MO.3(1)	Y	N
19.	What is the cohesion value of each function and module? MO.3(2)		
20.	Do module descriptions include identification of module interfaces? SI.1(9)	Y	N
21.	Is module designed in top down fashion? SI.1(1)	Y	N
22.	Number of functions performed. FS.1(1)		

3.2 TOLERANCE (RELIABILITY, SURVIVABILITY)

1.	When an error condition is detected, is it passed to calling module? AM.1(3)	Y	N
2.	Have numerical techniques being used in algorithm been analyzed with regards to accuracy requirements? AY.1(4)	Y	N
3.	Are values of inputs range tested? AM.2(2)	Y	N
4.	Are conflicting requests and illegal combinations identified and checked? AM.2(3)	Y	N

AD-A137 956 SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS 2/3
VOLUME 2 GUIDEBOOK F..(U) BOEING AEROSPACE CO SEATTLE
WA T P BOWEN ET AL. JUL 83 RADC-TR-83-175-VOL-2
UNCLASSIFIED F30602-80-C-0330 F/G 9/2 NL


1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

METRIC WORKSHEET 3	SYSTEM NAME:	DATE:
DESIGN/MODULE LEVEL	MODULE NAME:	INSPECTOR:

5.	Is there a check to see if all necessary data is available before processing begins? AM.2(5)	Y N
6.	Is all input checked, reporting all errors, before processing begins? AM.2(4)	Y N
7.	Are loop and multiple transfer index parameters range tested before use? AM.3(2)	Y N
8.	Are subscripts range tested before use? AM.3(3)	Y N
9.	Are outputs checked for reasonableness before processing continues? AM.3(4)	Y N
10.	Are checksums computed and transmitted with all messages? AM.6(3)	Y N
11.	Are checksums computed and compared upon message reception? AM.6(4)	Y N
12.	Are the number of transmission retries limited? AM.6(5)	Y N
13.	Are adjacent nodes checked periodically for operational status? AM.7(3)	Y N
14.	Are there alternate strategies for message routing? AM.7(4)	Y N
15.	Have accuracy requirements been budgeted to modules? AY.1(6)	Y N
3.3	**OPTIMIZATION (EFFICIENCY)**	
1.	Are specific performance requirements (storage and routine) allocated to this module? EF.1(1)	Y N
2.	Which category does processing fall in: EF.2 Real-time On-line Time-constrained Non-time critical	
3.	How many loops have non-loop dependent statements? EF.2(1)	
4.	Is bit/byte packing/unpacking performed in loops? EF.2(5)	Y N
5.	Is data indexed or reference efficiently? EF.3(5)	Y N
6.	Is performance optimizing compiler/assembly language used? EF.2(2)	Y N
3.4	**COMPLETENESS (CORRECTNESS)**	
1.	Is there a matrix relating functional requirements to the module which implements those requirements? TR.1(1)	Y N
2.	Can you clearly distinguish inputs, outputs, and the function being peformed? CP.1(1)	Y N
3.	How many data references are not defined, computed, or obtained from an external source? CP.1(2)	
4.	Are all conditions and processing defined for each decision point? CP.1(5)	Y N
5.	How many problem reports have been recorded for this module? CP.1(7)	

METRIC WORKSHEET 3	SYSTEM NAME:	DATE:
DESIGN/MODULE LEVEL	MODULE NAME:	INSPECTOR:

6. Number of problem reports still outstanding CP.1(7)
7. Profile of Problem Reports: (Number of Following Types)

a. Computational	h. Routine/System Interface	p. Recurrent Errors
b. Logic		q. Documentation
c. Input/Output	i. Tape Processing	r. Requirement Compliance
d. Data Handling	j. User Interface	s. Operator
e. System/OS Support	k. Data Base Interface	t. Questions
f. Configuration	l. User Requested Changes	u. Hardware
g. Routine/Routine Interface	m. Preset Data	v. Network Protocol
	n. Global Variable Definition	w. Communication Routing

3.5 REFERENCES (MAINTAINABILITY, FLEXIBILITY, VERIFIABILITY, PORTABILITY, REUSABILT INTEROPERABILITY, EXPANDABILITY, SURVIVABILITY)

1.	Number of references to system library routines, utilities or other system provided facilities ID.1(1)	
2.	Is a common, standard subset of programming language to be used? ID.1(2)	Y N
3.	Is the programming language available in other machines? ID.2(1)	Y N
4.	Number of input/output actions. ID.2(2)	
5.	Number of calling sequence parameters MO.2(3)	
6.	How many calling sequence parameters are control variables? MO.2(3)	
7.	Is input passed as calling sequence parameters MO.2(4)	Y N
8.	Is output passed back to calling module? MO.2(5)	Y N
9.	Is control returned to calling module? MO.2(6)	Y N
10.	Is temporary storage not shared with other modules? MO.2(7)	Y N
11.	Does the module associate with database system? AI.1(1)	Y N
12.	Number of the domains in system AI.5(1)	
13.	Number of the domains algorithm works for in system AI.5(1)	
14.	Is the algorithm certification available? AI.5(3)	Y N
15.	Is the algorithm test data available? AI.5(4)	Y N

METRIC WORKSHEET 3	SYSTEM NAME:	DATE:
DESIGN/MODULE LEVEL	MODULE NAME:	INSPECTOR:

3.6 CHANGEABILITY (FLEXIBILITY, REUSABILITY, EXPANDABILITY, INTEROPERABILITY)

1.	Is logical processing independent of storage specification? AG.1(1)	Y	N
2.	Percent of memory allocation uncommitted. AG.1(2)		
3.	Are accuracy, convergence, or timing attributes and limitations parametric? AG.2(1)	Y	N
4.	Is module table driven? AG.2(2)	Y	N
5.	Percent of cycle time allocation uncommitted. AG.2(3)		
6.	I/O channel time allocation uncommitted. AG.3(1)		
7.	Communication channel time allocation uncommitted. AG.3(2)		
8.	Does the module not mix input, output and processing functions in same module? GE.2(1)	Y	N
9.	Number of machine dependent functions performed? GE.2(2)		
10.	Is processing not data volume limited? GE.2(3)	Y	N
11.	Is processing not data value limited? GE.2(4)	Y	N

3.7 SYSTEM INTERFACES (SURVIVABILITY)

1.	Estimated lines of interface code. AU.1(2)	
2.	Estimated lines of source code. AU.1(2)	
3.	Estimated number of interface modules. AU.1(3)	
4.	Estimated time engaged in communication. AU.1(4)	

3.8 CONSISTENCY (CORRECTNESS, RELIABILITY, MAINTAINABILITY)

1.	Does the design representation comply with established standards CS.1(1)	Y	N
2.	Do input/output references comply with established standards CS.1(3)	Y	N
3.	Do calling sequences comply with established standards CS.1(2)	Y	N
4.	Is error handling done according to established standards CS.1(4)	Y	N
5.	Are variables named according to established standards CS.2(2)	Y	N
6.	Are global variables used as defined globally CS.2(3)	Y	N
7.	Does the data usage representation comply with established standards? CS.2(1)	Y	N

METRIC WORKSHEET 3	SYSTEM NAME:	DATE:
DESIGN/MODULE LEVEL	MODULE NAME:	INSPECTOR:

3.9 FUNCTIONAL CATEGORIZATION

Categorize function performed by this module according to following:

CONTROL - an executive module whose prime function is to invoke other modules.

INPUT/OUTPUT - a module whose prime function is to communicate data between the computer and either the user or another computer.

PRE/POSTPROCESSOR - a module whose prime function is to prepare data for or after the invocation of a computation or data management module.

ALGORITHM - a module whose prime function is computation.

DATA MANAGEMENT - a module whose prime function is to control the flow of data within the computer.

SYSTEM - a module whose function is the scheduling of system resources for other modules.

COMMUNICATION - a module whose prime function is to manage messasge routing between nodes.

NETWORK MANAGEMENT - a module whose prime function is to monitor and control network-level resources.

3.10 INSPECTOR'S COMMENTS

Make any specific or general comments about the quality observed while applying this checklist.

METRIC WORKSHEET 4	SYSTEM NAME:	DATE:
SOURCE CODE/MODULE LEVEL	MODULE NAME:	INSPECTOR:

4.1 STRUCTURE (RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, PORTABILIT REUSABILITY, EXPANDABILITY, CORRECTNESS)

1.	Number of lines excluding comments SI.4(2)	
2.	Number of declarative statements SI.4(9)	
3.	Number of data manipulation statements SI.4(9)	
4.	Number of statement labels (Do not count format statements SI.4(6)	
5.	Number of entrances into module SI.1(5)	
6.	Number of exits from module SI.1(5)	
7.	Maximum nesting level SI.4(7)	
8.	Number of decision points (IF, WHILE, REPEAT, DO, CASE) SI.3(1)	
9.	Number of sub-decision points. SI.3(1)	
10.	Number of conditional branches (computed go to) SI.4(8)	
11.	Number of unconditional branches (GOTO, ESCAPE) SI.4(8)	
12.	Number of loops (WHILE, DO) SI.4(3,4)	
13.	Number of loops with jumps out of loop SI.4(3)	
14.	Number of loop indices that are modified SI.4(4)	
15.	Number of constructs that perform module modifications (SWITCH, ALTER) SI.4(5) (Also see 4.5, MO.2(2))	
16.	Number of negative or complicated compound boolean expressions SI.4(2)	
17.	Is a structured language used SI.2(1)	Y N
18.	Is flow top to bottom (are there no backward branching GOTOs) SI.4(1)	Y N
19.	Is code written according to a programming standard? SI.4(13)	Y N
20.	Are macros and subroutines used to avoid repeated and redundant code? SI.4(14)	Y N
21.	Number of data items used to specify the interface. SC.1(1)	
22.	Number of data items passed implicitly across interface via common global data without adequate comments. SC.1(2)	
23.	Number of nesting levels in interface. SC.1(3)	
24.	Number of interface data items with negative qualification. SC.1(4)	
25.	Number of data items passed across module interface. SC.1(5)	
26.	Does the module have comments about the common control blocks, common data blocks and global variable names in module interface? SC.1(6)	Y N
27.	Does the module modify other modules? SC.1(7)	Y N
28.	Number of possible unique execution paths. SC.2(1)	

METRIC WORKSHEET 4	SYSTEM NAME:	DATE:
SOURCE CODE/MODULE LEVEL	MODULE NAME:	INSPECTOR:

29.	Number of IF statements. SC.2(2)	
30.	Number of function CALLs. SC.2(3)	
31.	Number of control variables used to direct execution path selection. SC.2(4)	
32.	Number of DO groups. SC.2(5)	
33.	Does the module have code comments about calling what modules and called by what modules? SC.2(6)	Y N
34.	Does the module share temporary storage with other modules? SC.3(2)	Y N
35.	Does the module have mixed database-management and storage-management routines? SC.3(3)	Y N
36.	Average number of formal parameters in each routine. SC.4(1)	
37.	Average number of common global variables used in each module. SC.4(2)	
38.	Number of global variables modified by one routine and referenced by another routines. SC.4(3)	
39.	Does the module connect to other modules with functional name? SC.4(4)	Y N
40.	Does the module communicate with other modules by passing control elements? SC.4(5)	Y N
41.	Number of machine level language statements. AI.3(3)	
42.	Does the module with logical processing depend on data storage specification and requirement? AI.2(4)	Y N
43.	Does the program compute the same value more than once? SC.5(1)	Y N
44.	Does the program insert a statement which never needs to be executed? SC.5(2)	Y N
45.	Does the program maintain a constant meaning for each variable? SC.5(3)	Y N
46.	Does the program use the unnecessary intermediate variables? SC.5(4)	Y N

4.2 TOLERANCE (RELIABILITY, SURVIVABILITY)

1.	Are loop and multiple transfer index parameters range tested before use? AM.3(2)	Y N
2.	Are subscript values range tested before use? AM.3(3)	Y N
3.	When an error condition occurs, is it passed to the calling module? AM.1(3)	Y N
4.	Are the results of a computation checked before outputting or before processing continues? AM.3(4)	Y N

METRIC WORKSHEET 4	SYSTEM NAME:	DATE:
SOURCE CODE/MODULE LEVEL	MODULE NAME:	INSPECTOR:

4.3 OPTIMIZATION (EFFICIENCY)

1.	Number of mix mode expressions? EF.3(3)	
2.	How many variables are initialized when declared? EF.3(2)	
3.	How many loops have non-loop dependent statements in them? EF.2(1)	
4.	Do loops have bit/byte packing/unpacking? EF.2(5), EF.4(6)	Y N
5.	How many compound expressions defined more than once? EF.2(3)	

4.4 CONCISENESS (MAINTAINABILITY) - SEE METRIC EXPLANATIONS

1.	Number of operators CO.1(1)	
2.	Number of unique operators CO.1(1)	
3.	Number of Operands CO.1(1)	
4.	Number of unique operands CO.1(1)	

4.5 REFERENCES (MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, PORTABILITY, REUSABILIT INTEROPERABILITY, EXPANDABILITY, SURVIVABILITY)

1.	Number of calls to other modules MO.2(1)	
2.	Number of references to system library routines, utilities, or other system provided functions ID.1(1)	
3.	Number of calling sequence parameters MO.2(3)	
4.	How many elements in calling sequences are not parameters? MO.2(3)	
5.	How many of the calling parameters (input) are control variables? MO.2(3)	
6.	How many parameters passed to or from other modules are not defined in this module? MO.2(3)	
7.	Is input data passed as parameter? MO.2(4)	Y N
8.	Is output data passed back to calling module? MO.2(5)	Y N
9.	Is control returned to calling module? MO.2(6)	Y N
10.	Number of lines of code? MO.2(2)	

4.6 CHANGEABILITY (FLEXIBILITY, INTEROPERABILITY, REUSABILITY, EXPANDABILITY)

1.	Is module table driven? AG.2(2)	Y N
2.	Are there any limits to data values that can be processed? GE.2(4)	Y N

METRIC WORKSHEET 4	SYSTEM NAME:	DATE:
SOURCE CODE/MODULE LEVEL	MODULE NAME:	INSPECTOR:

3.	Are there any limits to amounts of data that can be processed? GE.2(3)	Y N
4.	Are accuracy, convergence and timing attributes parametric? AG.2(1)	Y N
5.	Amount of memory used. AG.1(2)	
6.	Does the module allow for modifying resource utilization? DA.3(2)	Y N
7.	Does the module have comments about functional descriptions? FS.1(2)	Y N
8.	Does the module have comments about algorithm descriptions? AI.5(5)	Y N
9.	Does the module have the selected computation or output features? DA.3(1)	Y N

4.7 INPUT/OUTPUT (RELIABILITY, PORTABILITY, REUSABILITY, SURVIVABILIT INTEROPERABILITY)

1.	Number of input statments ID.2(2)	
2.	Number of output statements ID.2(2)	
3.	Are inputs range-tested (for inputs via calling sequences, global data, and input statements) AM.2(2)	Y N
4.	Are possible conflicts or illegal combinations in inputs checked? AM.2(3)	Y N
5.	Is there a check to determine if all data is available prior to processing? AM.2(5)	Y N
6.	Is all input checked, reporting all errors, before processing begins? AM.2(4)	Y N
7.	Number of lines of interface code. AU.1(2)	
8.	Number of modules with interface code. AU.1(3)	
9.	Are the input/output formats well defined? FS.2(3)	Y N

4.8 SELF-DESCRIPTIVENESS (MAINTAINABILITY, FLEXIBILITY, VERIFIABILITY, PORTABILIT REUSABILITY)

1.	Number of lines of source code SD.1(1)	
2.	Number of non-blank lines of comments SD.1(1)	
3.	Are there prologue comments provided containing information about the function, author, version number, date, inputs, outputs, assumptions and limitations? SD.2(1)	Y N
4.	Is there a comment which indicates what itemized requirement is satisfied by this module? SD.2(1)	Y N
5.	How many decision points and transfers of control are not commented? SD.2(3)	
6.	Is all machine language code commented? SD.2(4)	Y N
7.	Are non-standard HOL statements commented? SD.2(5)	Y N
8.	How many declared variables are not described by comments? SD.2(6)	

METRIC WORKSHEET 4	SYSTEM NAME:	DATE:
SOURCE CODE/MODULE LEVEL	MODULE NAME:	INSPECTOR:

9.	Are variable names (mnemonics) descriptive of the physical or functional property they represent? SD.3(2)	Y	N
10.	Do the comments do more than repeat the operation? SD.2(7)	Y	N
11.	Is the code logically blocked and indented? SD.3(3)	Y	N
12.	Number of lines with more than 1 statement. SD.3(4)		
13.	Number of continuation lines. SD.3(4)		
14.	Are comments set off from code in a uniform manner? SD.2(2)	Y	N
15.	Is this module free of machine level language statements? SD3(1) (Also see 4.1, AI.3(3))	Y	N
16.	Is the module in the standard format organization? SD.3(5)	Y	N
17.	Does the module use the language keywords? SD.3(6)	Y	N

4.9 DATA (CORRECTNESS, RELIABILITY, MAINTAINABILITY, VERIFIABILITY, EFFICIENC FLEXIBILITY, REUSABILITY, EXPANDABILITY)

1.	Number of local variables SI.4(10)		
2.	Number of global variables SI.4(10)		
3.	Number of global variables renamed EF.4(3)		
4.	How many variables are used for more than one purpose? CS.2(3)		
5.	Number of executable statements. SI.4(11)		
6.	Number of variables used? SI.4(11)		
7.	Does each variable have single use? SI.4(12)	Y	N
8.	Number of occurrences of uncommon unit operations EF.3(4)		
9.	Does the module have comments about input data value range and their default conditions? SD.2(8)	Y	N
10.	Does the module have the code comments about data items used? AI.2(5)	Y	N
11.	How many data items are described parametrically? AI.2(1)		
12.	How many data items could be described parametrically? AI.2(1)		
13.	Does each module have comments about global, local parameter variables? AI.2(2)	Y	N

4.10 INDEPENDENCE (PORTABILITY, REUSABILITY, INTEROPERABILITY)

1.	Is code independent of word and character size? ID.2(3)	Y	N
2.	Is a common, standard subset of programming language used? ID.1(2)	Y	N
3.	Is data representation machine independent? ID.2(4)	Y	N

METRIC WORKSHEET 4	SYSTEM NAME:	DATE:
SOURCE CODE/MODULE LEVEL	MODULE NAME:	INSPECTOR:

4.11 DYNAMIC MEASUREMENTS (EFFICIENCY, RELIABILITY, FLEXIBILITY, EXPANDABILITY, SURVIVABILITY)

		Y	N
1.	During execution are outputs within accuracy tolerances? AY.1(5)		
2.	During module/development testing, what was run time? AG.2(3)		
3.	Complete memory map for execution of this module EF.4(4) Size (words of memory) APPLICATION SYSTEM DATA OTHER		
4.	During execution how many data items were referenced but not modified? EF.3(6)		
5.	During execution how many data items were modified? EF.3(7)		
6.	Amount of I/O channel capacity used. AG.3(1)		
7.	Amount of communication channel capacity used. AG.3(2)		
8.	Time engaged in communication. AU.1(4)		
9.	Module linkage time EF.2(6)		
10.	Module execution time EF.2(6)		
11.	OS linkage time EF.2(7)		
12.	OS execution time EF.2(7)		

4.12 INSPECTORS COMMENTS

Make any general or specific comments that relate to the quality observed while applying this checklist.

APPENDIX B
METRIC TABLES

Appendix B contains the metric tables which are used for calculating values for metrics and metric elements. The tables are organized alphabetically by quality criteria name and numerically by metric acronym. A summary of the metric tables and a correlation to metric worksheets are shown on the next several pages.

Each metric table identifies the quality criteria, the metric, and the metric element and references the applicable quality factors. Formulas are stated, where appropriate, to calculate values for metric elements and for metrics. Each metric element is cross-referenced to the software development phase during which it is applicable and to the appropriate worksheet and worksheet section(s) (see Appendix A, Metric Worksheets). The worksheet cross-reference is by a decimal number scheme. If, for example, 1.2 is called out, this refers to Metric Worksheet 1, Section 2. A cross-reference enclosed in parentheses indicates a reapplication of the metric element during a subsequent development phase.

Each metric in the tables is identified by a type code: an (a) following the metric name identifies an anomaly detecting metric, and a (p) identifies a predictive metric. If a normalization function has been established for a quality factor but the metric is not included, it is because the metric did not illustrate sufficient correlation with the operational history. In lieu of inclusion in the normalization function, some metrics are maintained as strictly anomaly-detecting metrics; they are felt to identify or assist in identification of problems which should be and are typically corrected immediately to enhance the quality of the product.

The contents of this appendix are based on the results of this contract, "Quality Metrics for Distributed Systems", F30602-80-C-0330 and the results of contract F30603-80-C-0265, "Software Interoperability and Reusability". This appendix includes a refinement and reorganization of metric table information initially defined in RADC-TR-77-369 and RADC-TR-80-109.

METRIC TABLES SUMMARY

CRITERIA	ACRONYM	METRICS
ACCURACY	AY.1	ACCURACY CHECKLIST
ANOMALY MANAGEMENT	AM.1	ERROR TOLERANCE/CONTROL CHECK-LIST
	AM.2	IMPROPER INPUT DATA CHECKLIST
	AM.3	COMPUTATIONAL FAILURES CHECKLIST
	AM.4	HARDWARE FAULTS CHECKLIST
	AM.5	DEVICE ERRORS CHECKLIST
	AM.6	COMMUNICATION ERRORS CHECKLIST
	AM.7	NODE/COMMUNICATIONS FAILURES CHECKLIST
APPLICATION INDEPENDENCE	AI.1	DATA BASE SYSTEM INDEPENDENCE
	AI.2	DATA STRUCTURE
	AI.3	ARCHITECTURE STANDARDIZATION
	AI.4	MICROCODE INDEPENDENCE
	AI.5	ALGORITHM
AUGMENTABILITY	AG.1	DATA STORAGE EXPANSION MEASURE
	AG.2	COMPUTATION EXTENSIBILITY MEASURE
	AG.3	CHANNEL EXTENSIBILITY MEASURE
	AG.4	DESIGN EXTENSIBILITY CHECKLIST
AUTONOMY	AU.1	INTERFACE COMPLEXITY MEASURE
	AU.2	SELF-SUFFICIENCY CHECKLIST
COMMONALITY	CL.1	COMMUNICATIONS COMMONALITY CHECKLIST
	CL.2	DATA COMMONALITY CHECKLIST
	CL.3	COMMON VOCABULARY CHECKLIST
COMMUNICATIVENESS	CM.1	USER INPUT INTERFACE MEASURE
	CM.2	USER OUTPUT INTERFACE MEASURE
COMPLETENESS	CP.1	COMPLETENESS CHECKLIST
CONCISENESS	CO.1	HALSTEAD'S MEASURE
CONSISTENCY	CS.1	PROCEDURE CONSISTENCY MEASURE
	CS.2	DATA CONSISTENCY MEASURE
DISTRIBUTEDNESS	DI.1	DESIGN STRUCTURE CHECKLIST
DOCUMENT ACCESSIBILITY	DA.1	ACCESS NO-CONTROL
	DA.2	WELL-STRUCTURED DOCUMENTATION
	DA.3	SELECTIVE USABILITY

METRIC TABLES SUMMARY

CRITERIA	ACRONYM	METRICS
EFFECTIVENESS	EF.1	PERFORMANCE REQUIREMENTS
	EF.2	ITERATIVE PROCESSING EFFICIENCY MEASURE
	EF.3	DATA USAGE EFFICIENCY MEASURE
	EF.4	STORAGE EFFICIENCY MEASURE
FUNCTIONAL OVERLAP	FO.1	FUNCTIONAL OVERLAP MEASURE
FUNCTIONAL SCOPE	FS.1	FUNCTION SPECIFICITY
	FS.2	FUNCTION COMMONALITY
	FS.3	FUNCTION COMPLETENESS
GENERALITY	GE.1	MODULE REFERENCE BY OTHER MODULES
	GE.2	IMPLEMENTATION FOR GENERALITY CHECKLIST
INDEPENDENCE	ID.1	SOFTWARE SYSTEM INDEPENDENCE MEASURE
	ID.2	MACHINE INDEPENDENCE MEASURE
MODULARITY	MO.2	MODULAR IMPLEMENTATION MEASURE
	MO.3	MODULAR DESIGN MEASURE
OPERABILITY	OP.1	OPERABILITY CHECKLIST
RECONFIGURABILITY	RE.1	RESTRUCTURE CHECKLIST
SELF-DESCRIPTIVENESS	SD.1	QUANTITY OF COMMENTS
	SD.2	EFFECTIVENESS OF COMMENTS MEASURE
	SD.3	DESCRIPTIVENESS OF LANGUAGE MEASURE
SIMPLICITY	SI.1	DESIGN STRUCTURE MEASURE
	SI.2	STRUCTURED LANGUAGE OR PRE-PROCESSOR
	SI.3	DATA AND CONTROL FLOW COMPLEXITY MEASURE
	SI.4	CODING SIMPLICITY MEASURE
SPECIFICITY	SP.1	SCOPE OF FUNCTION MEASURE
SYSTEM ACCESSIBILITY	SA.1	ACCESS CONTROL CHECKLIST
	SA.2	ACCESS AUDIT CHECKLIST

METRIC TABLES SUMMARY

CRITERIA	ACRONYM	METRICS
SYSTEM CLARITY	SC.1 SC.2 SC.3 SC.4 SC.5	INTERFACE COMPLEXITY PROGRAM FLOW COMPLEXITY APPLICATION FUNCTIONAL COMPLEXITY COMMUNICATION COMPLEXITY STRUCTURE CLARITY
SYSTEM COMPATIBILITY	SY.1 SY.2 SY.3 SY.4 SY.5	COMMUNICATION COMPATIBILITY CHECKLIST DATA COMPATIBILITY CHECKLIST HARDWARE COMPATIBILITY CHECKLIST SOFTWARE COMPATIBILITY CHECKLIST DOCUMENTATION FOR OTHER SYSTEM
TRACEABILITY	TR.1	CROSS REFERENCE
TRAINING	TN.1	TRAINING CHECKLIST
VIRTUALITY	VR.1	SYSTEM/DATA INDEPENDENCE CHECK-LIST
VISIBILITY	VS.1 VS.2 VS.3	MODULE TESTING MEASURE INTEGRATION TESTING MEASURE SYSTEM TESTING MEASURE

METRIC WORKSHEET CORRELATION

QUALITY FACTORS														PHASES				
CORRECTNESS	RELIABILITY	EFFICIENCY	INTEGRITY	USABILITY	SURVIVABILITY	MAINTAINABILITY	VERIFIABILITY	FLEXIBILITY	PORTABILITY	REUSABILITY	INTEROPERABILITY	EXPANDABILITY	CRITERIA/ METRIC	REQUIREMENTS ANALYSIS	PRELIMINARY DESIGN	DETAILED DESIGN	IMPLEMENTATION	TEST & INTEGRATION
	X												ACCURACY					
													AY.1	1.2	2.2	3.2	(3.2),4.11	
	X				X								ANOMALY MANAGE-MENT					
													AM.1		2.2	3.2	4.2	2.2
													AM.2	1.2		3.2	4.7	
													AM.3	1.2		3.2	4.2	
													AM.4	1.2	2.2			2.2
													AM.5	1.2	2.2			2.2
													AM.6	1.2	2.2	3.2		2.2
													AM.7	1.2	2.2	3.2		2.2
										X			APPLICATION INDEPENDENCE					
													AI.1		2.5	3.5		2.5
													AI.2		2.8		4.1,4.9	2.8
													AI.3		2.5		4.1	2.5
													AI.4		2.5			2.5
													AI.5		2.5	3.5	4.6	2.5
											X	X	AUGMENT-ABILITY					
													AG.1	1.6	2.6	3.6	(3.6),4.6	2.6
													AG.2	1.6	2.6	3.6	4.6,4.11	2.6
													AG.3	1.6	2.6	3.6	4.11	2.6
													AG.4	1.6	2.6			2.6
					X								AUTONOMY					
													AU.1	1.7	2.7	3.7	4.7,4.11	2.7
													AU.2	1.7	2.7			2.7
											X		COMMONALITY					
													CL.1	1.7	2.7			2.7
													CL.2	1.7	2.7			2.7
													CL.3	1.10				
				X							X		COMMUNICATIVE-NESS					
													CM.1	1.9	2.9			2.9
													CM.2	1.9	2.9			2.9
X													COMPLETENESS					
													CP.1	1.4	2.4	3.4	(3.4)	2.4
						X							CONCISENESS					
													CO.1				4.4	
X	X					X							CONSISTENCY					
													CS.1			3.8	(3.8)	
													CS.2	1.8	2.8	3.8	(3.8),4.9	2.8
					X								DISTRIBUTED-NESS					
													DI.1	1.1,1.8	2.1,2.8	3.1		2.1,2.8
										X			DOCUMENT ACCESSIBILITY					
													DA.1	1.11				
													DA.2	1.11				
													DA.3	1.11			4.6	
		X											EFFECTIVENESS					
													EF.1	1.3	2.3	3.3		
													EF.2		2.3	3.3	4.3,4.11	2.3
													EF.3		2.3	3.3	(3.3) 4.3, 4.9, 4.11	2.3
													EF.4		2.3		4.3,4.9, 4.11	2.3
											X		FUNCTIONAL OVERLAP					
													FO.1	1.13				

METRIC WORKSHEET CORRELATION

QUALITY FACTORS														PHASES				
CORRECTNESS	RELIABILITY	EFFICIENCY	INTEGRITY	USABILITY	SURVIVABILITY	MAINTAINABILITY	VERIFIABILITY	FLEXIBILITY	PORTABILITY	REUSABILITY	INTEROPERABILITY	EXPANDABILITY	CRITERIA/ METRIC	REQUIREMENTS ANALYSIS	PRELIMINARY DESIGN	DETAILED DESIGN	IMPLEMENTATION	TEST & INTEGRATION
										X			FUNCTIONAL SCOPE					
													FS.1			3.1	4.6	
													FS.2	1.5			4.7	
													FS.3	1.5				
								X		X		X	GENERALITY					
													GE.1		2.6			2.6
													GE.2			3.6	4.6	
									X	X	X		INDEPENDENCE					
													ID.1			3.5	4.5,4.10	
													ID.2			3.5	(3.5),4.7 4.10	
					X	X	X	X	X	X	X	X	MODULARITY					
													MO.2		2.1	3.5	(3.5)4.5	
													MO.3	1.1	2.1	3.1		
				X									OPERABILITY					
													OP.1	1.9	2.9			2.9
					X								RECONFIGUR-ABILITY					
													RE.1	1.7,1.8	2.7,2.8			2.7,2.8
						X	X	X	X	X			SELF-DESCRIPTIVENESS					
													SD.1				4.8	
													SD.2				4.8,4.9	
													SD.3				4.8	
X	X					X	X	X		X		X	SIMPLICITY					
													SI.1	1.1	2.1,2.8	3.1	4.1	2.1,2.8
													SI.2				4.1	
													SI.3			3.1	4.1	
													SI.4	1.1		3.1	4.1,4.9	
X							X					X	SPECIFICITY					
													SP.1			3.1		
			X										SYSTEM ACCESSIBILITY					
													SA.1	1.12	2.12			2.12
													SA.2	1.12	2.12			2.12
										X			SYSTEM CLARITY					
													SC.1				4.1	
													SC.2				4.1	
													SC.3		2.1		4.1	2.1
													SC.4				4.1	
													SC.5				4.1	
											X		SYSTEM COMPATIBILITY					
													SY.1		2.11			2.11
													SY.2		2.11			2.11
													SY.3		2.11			
													SY.4		2.11			
													SY.5	1.11				
X													TRACEABILITY					
													TR.1	1.4	2.4	3.4		
				X									TRAINING					
													TN.1		2.9			2.9
			X	X								X	VIRTUALITY					
													VR.1	1.8	2.1, 2.8			2.1,2.8
				X		X	X						VISIBILITY					
													VS.1		2.10			2.10
													VS.2		2.10			2.10
													VS.3		2.10			2.10

( ) = Reapplication of Metric During Subsequent Phase

CRITERIA: ACCURACY	FACTOR(S): RELIABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AY.1 ACCURACY CHECKLIST (a): (1) Error analysis performed and budgeted to module.	1.2									
(2) A definitive statement of requirement for accuracy of inputs, outputs, processing, and constants.	1.2									
(3) Sufficiency of math library.			2.2							
(4) Sufficiency of numerical methods.					3.2		(3.2)			
(5) Execution outputs within tolerances.							4.11			
(6) Accuracy requirements budgeted to functions/modules			2.2		3.2					
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: ANOMALY MANAGEMENT	FACTOR(S): RELIABILITY, SURVIVABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AM.1 ERROR TOLERANCE/CONTROL CHECKLIST (p): (1) Any concurrent processing centrally controlled.			2.2						2.2	
(2) Errors should be fixable and processing continued. (Errors fixable/total error conditions)				2.2					2.2	
(3) When an error condition is detected, it should be passed up to calling routine.					3.2		4.2			
(4) Any parallel processing centrally controlled.			2.2						2.2	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: ANOMALY MANAGEMENT	FACTOR(S): RELIABILITY, SURVIVABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AM.2 IMPROPER INPUT DATA CHECKLIST (p): (1) A definitive statement of requirement for error tolerance of input data.	1.2									
(2) Range of values (reasonableness) for items specified and checked.					3.2		4.7			
(3) Conflicting requests and illegal combinations identified and checked.					3.2		4.7			
(4) All input is checked before processing begins					3.2		4.7			
(5) Determination that all data is available prior to processing.					3.2		4.7			
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: ANOMALY MANAGEMENT	FACTOR(S): RELIABILITY, SURVIVABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AM.3 COMPUTATIONAL FAILURES CHECKLIST (p): (1) A definitive statement of requirement for recovery from computational failures.	1.2									
(2) Loop and multiple transfer index parameters range tested before use.					3.2		4.2			
(3) Subscript checking.					3.2		4.2			
(4) Critical output parameters reasonableness checked during processing.					3.2		4.2			
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: ANOMALY MANAGEMENT	FACTOR(S): RELIABILITY, SURVIVABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AM.4 HARDWARE FAULTS CHECKLIST (a): (1) A definitive statement of requirement for recovery from hardware faults.	1.2									
(2) Recovery from hardware faults (e.g., arithmetic faults, power failure, clock).			2.2						2.2	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: ANOMALY MANAGEMENT	FACTOR(S): RELIABILITY, SURVIVABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AM.5 DEVICE ERRORS CHECKLIST (a): (1) Definitive statement of requirement for recovery from device errors.	1.2									
(2) Recovery from device errors.			2.2						2.2	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: ANOMALY MANAGEMENT	FACTOR(S): RELIABILITY, SURVIVABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AM.6 COMMUNICATION ERRORS CHECKLIST (a): (1) A definitive statement of requirements for recovery from communication errors.	1.2									
(2) Provisions for recovery from communication errors.			2.2						2.2	
(3) Check sums computed and transmitted with all messages.					3.2					
(4) Check sums computed and compared upon message reception.					3.2					
(5) Transmission retries limited.					3.2					
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: ANOMALY MANAGEMENT	FACTOR(S): RELIABILITY, SURVIVABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AM.7 NODE/COMMUNICATIONS FAILURES CHECKLIST (a): (1) A definitive statement of requirements for rcovery from node/communication failures.	1.2									
(2) Provisions for recovery from node/communication failures.			2.2						2.2	
(3) Adjacent nodes checked for operational status.					3.2					
(4) Alternate strategies for message routing.					3.2					
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: APPLICATION INDEPENDENCE	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AI.1 DATABASE SYSTEM INDEPENDENCE (p): (1) Free from database system reference $1 - \frac{\text{\# modules with DB system reference}}{\text{total \# modules}}$				2.5	3.5					2.5
METRIC VALUE = Same as line above										

CRITERIA: APPLICATION INDEPENDENCE	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AI.2 DATA STRUCTURE (p): (1) Data in parameter list, data structure described parametrically $\frac{\text{\# data structure with parametric definition}}{\text{\# data structure that could be parametrized}}$								4.9		
(2) Data communicated through common storage region with adequate comments							4.9			
(3) Control of database structures, both global and local, i.e., all data centrally controlled and symbolically defined and referenced			2.8						2.8	
(4) Logical processing independent of data storage specification and requirement $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.1		
(5) Each module has code comments about data items description including global & parameter input/output and local variables $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.9		
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: APPLICATION INDEPENDENCE	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AI.3 ARCHITECTURE STANDARDIZATION (p): (1) Module is free from computer architecture reference $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$				2.5						2.5
(2) Module is in standard 32 bits computer architecture (Nebula) $\frac{\text{\# modules in standard architecture}}{\text{total \# modules}}$				2.5						2.5
(3) Code statements are free from machine architecture $1 - \frac{\text{\# of lines of machine language statements}}{\text{Total locations}}$							4.1			
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: APPLICATION INDEPENDENCE	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AI.4 MICROCODE INDEPENDENCE (p): (1) Number of modules used microcode instruction $1 - \frac{\text{\# modules use microcode}}{\text{total \# modules}}$				2.5						2.5
METRIC VALUE = same as entry above										

CRITERIA: APPLICATION INDEPENDENCE	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AI.5 ALGORITHM (p): (1) Valid range $\frac{\text{\# domains algorithm works for}}{\text{\# possible domains}}$						3.5				
(2) Is the algorithm table driven $1 - \frac{\text{\# module not with table driven algorithm}}{\text{total \# modules}}$				2.5						2.5
(3) Is the algorithm certification available					3.5					
(4) Is the algorithm test data available					3.5					
(5) Each module has code comments about algorithm description $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.6		
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: AUGMENTABILITY	FACTOR(S): INTEROPERABILITY, EXPANDABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AG.1 DATA STORAGE EXPANSION MEASURE (a): (1) Logical processing independent of storage specification/requirements (by module) $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$						3.6		(3.6)		
(2) Percent of memory capacity uncommitted $\frac{\text{Amount of memory uncommitted}}{\text{Total amount of available memory}}$	1.6			2.6		3.6		4.6		2.6
(3) Percent auxiliary storage capacity uncommitted $\frac{\text{Amount of auxiliary storage uncommitted}}{\text{Total amount of available auxiliary storage}}$	1.6			2.6						2.6
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: AUGMENTABILITY	FACTOR(S): INTEROPERABILITY, EXPANDABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AG.2 COMPUTATION EXTENSIBILITY MEASURE (a): (1) Accuracy, convergence, timing attributes which control processing are parametric $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$						3.6		4.6		
(2) Modules table driven $1 - \frac{\text{\# modules not table driven}}{\text{total \# modules}}$						3.6		4.6		
(3) Percent of speed capacity uncommitted $\frac{\text{Amount of cycle time uncommitted}}{\text{total processing time}}$	1.6			2.6		3.6		4.11		2.6
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: AUGMENTABILITY	FACTOR(S): INTEROPERABILITY, EXPANDABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AG.3 CHANNEL EXTENSIBILITY MEASURE (p): (1) Spare I/O channel capacity (by peripheral) $1 - \frac{\text{bits per second (peak) committed}}{\text{bits per second (peak) available}}$	1.6			2.6		3.6		4.11		2.6
(2) Spare communication channel capacity $1 - \frac{\text{bits per second (peak) committed}}{\text{bits per second (peak) available}}$	1.6			2.6		3.6		4.11		2.6
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: AUGMENTABILITY	FACTOR(S): INTEROPERABILITY, EXPANDABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AG.4 DESIGN EXTENSIBILITY CHECKLIST (p): (1) Processors, communication links, memory devices, and peripherals of a common vendor or model	1.6		2.6						2.6	
(2) Documentation reveals performanced price of software/system for enhancement trades.	1.6		2.6							
(3) Specifications identify new technology tradeoff areas for software.	1.6		2.6							
(4) Software specifications include requirements for the criteria of the quality factor expandability	1.6		2.6							
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: AUTONOMY	FACTOR(S): SURVIVABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AU.1 INTERFACE COMPLEXITY MEASURE (p): (1) Processes/functions separated as logical "wholes" to minimize interface complexity.	1.7		2.7							
(2) Interface code $1 - \frac{\text{lines of interface code}}{\text{total lines code}}$				2.7		3.7		4.7		
(3) Interface modules $1 - \frac{\text{\# modules with interface code}}{\text{total \# modules}}$				2.7		3.7		4.7		2.7
(4) Communication loading $1 - \frac{\text{\% time engaged in communication}}{100}$				2.7		3.7		4.11		2.7
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: AUTONOMY	FACTOR(S): SURVIVABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
AU.2 SELF-SUFFICIENCY CHECKLIST (p): (1) Software volatility - each CPU/system has separate power supply.	1.7		2.7							
(2) Each scheduling unit (i.e. executive, operating system) tests its own operation, communication links, memories, and peripherals	1.7		2.7						2.7	
(3) Software system includes word-processing capability	1.7		2.7						2.7	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: COMMONALITY	FACTOR(S): INTEROPERABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
CL.1 COMMUNICATIONS COMMONALITY CHECKLIST (a): (1) Definitive statement of requirement for communication with other systems	1.7		2.7							
(2) Protocol standards established and followed for network process control			2.7						2.7	
(3) Single module interface for input $\frac{1}{\text{\# modules used for input}}$				2.7						2.7
(4) Single module interface for output $\frac{1}{\text{\# modules used for output}}$				2.7						2.7
(5) Specific requirements for network process control	1.7									
(6) Specific requirements for user session control	1.7									
(7) Specific requirements for communication routing strategy	1.7									
(8) Protocol standards established and followed for user session control			2.7						2.7	
(9) Protocol standards established and followed for communication routing			2.7						2.7	

CRITERIA: COMMONALITY	FACTOR(S): INTEROPERABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
CL.1 COMMUNICATIONS COMMONALITY CHECKLIST (Continued) (10) Number of systems responding correctly to successfully complete handshaking $\frac{1}{\text{\# of systems}}$		1.7								
(11) Low time dependency on handshaking	1.7									
(12) No communication time dependency that effects system performance (1 = Yes, 0 = No)	1.7									
(13) Number of other systems this system will interface with $\frac{1}{\text{\# other systems}}$				2.7						
(14) No timing dependency on data freshness	1.7									
(15) Operating procedure known	1.7									
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: COMMONALITY	FACTOR(S): INTEROPERABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
CL.2 DATA COMMONALITY CHECKLIST (a): (1) Definitive statement for standard data representation for communication with other systems	1.7									
(2) Translation standards among representations established and followed			2.7						2.7	
(3) Single module to perform each translation $\frac{1}{\text{\# modules used to perform translation}}$				2.7						2.7
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: COMMONALITY	FACTOR(S): INTEROPERABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
CL.3 COMMON VOCABULARY CHECKLIST (a): (1) Same vocabulary used on both projects with identical meanings?	1.10									
METRIC VALUE = 1 (YES) 0 (NO)										

CRITERIA: COMMUNICATIVENESS	FACTOR(S): USABILITY, INTEROPERABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
CM.1 USER INPUT INTERFACE MEASURE (a): (1) Default values defined $\frac{\text{\# defaults}}{\text{total \# parameters}}$				2.9						2.9
(2) Input formats uniform $\frac{1}{\text{\# different input record formats}}$				2.9						2.9
(3) Each input record self identifying $1 - \frac{\text{\# that are not self identifying}}{\text{total \# input records}}$				2.9						2.9
(4) Input can be verified by user prior to execution			2.9						2.9	
(5) Input terminated by explicitly defined logical end of input			2.9						2.9	
(6) Provisions for specifying input from different media	1.9		2.9						2.9	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: COMMUNICATIVENESS	FACTOR(S): USABILITY, INTEROPERABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
CM.2 USER OUTPUT INTERFACE MEASURE (a): (1) Selective output controls	1.9		2.9						2.9	
(2) Outputs have unique descriptive user oriented labels			2.9						2.9	
(3) Outputs have user oriented units			2.9						2.9	
(4) Uniform output formats $\frac{1}{\#\text{ different output formats}}$				2.9						2.9
(5) Logical groups of output separated for user examination			2.9						2.9	
(6) Relationship between error messages and outputs is unambiguous			2.9						2.9	
(7) Provision for redirecting output to different media	1.9		2.9						2.9	
(8) Standard user interfaces for network information and data access	1.9		2.9						2.9	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\#\text{ applicable elements}}$										

CRITERIA: COMPLETENESS	FACTOR(S): CORRECTNESS									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
CP.1 COMPLETENESS CHECKLIST (a): (1) Unambiguous requirements/references for input, function and output	1.4		2.4		3.4		(3.4)			
(2) All data references defined, computed or obtained from an external source. $\frac{\text{data references defined}}{\text{total data references}}$		1.4				3.4		(3.4)		
(3) All defined functions used. $\frac{\text{defined function used}}{\text{total functions identified}}$		1.4		2.4						2.4
(4) All referenced functions defined. $\frac{\text{referenced functions defined}}{\text{total functions identified}}$		1.4		2.4						2.4
(5) All conditions and processing defined for each decision point.	1.4				3.4		(3.4)			
(6) All defined and referenced calling sequence parameters agree. $\frac{\text{parameters agree}}{\text{total parameters}}$		1.4		2.4						2.4
(7) All problem reports resolved. $\frac{\text{problem reports resolved}}{\text{total problem reports}}$	1.4			2.4		3.4		(3.4)		2.4
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: CONCISENESS	FACTOR(S): MAINTAINABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
CO.1 HALSTEAD'S MEASURE (by module)(p): (1) $1 - \frac{\text{module length calculated-module length observed}}{\text{module length calculated}}$								4.4		
METRIC VALUE = Same as entry above										

CRITERIA: CONSISTENCY	FACTOR(S): CORRECTNESS, RELIABILITY, MAINTAINABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
CS.1 PROCEDURE CONSISTENCY MEASURE (a): (1) Standard design representation $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$						3.8				
(2) Calling sequence conventions $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$						3.8		(3.8)		
(3) Input/output conventions $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$						3.8		(3.8)		
(4) Error handling conventions $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$						3.8		(3.8)		
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: CONSISTENCY	FACTOR(S): CORRECTNESS, RELIABILITY, MAINTAINABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
CS.2 DATA CONSISTENCY MEASURE (a): (1) Standard data usage representation $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$						3.8				
(2) Naming conventions $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$						3.8		(3.8)		
(3) Consistent global definitions $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$						3.8		4.9		
(4) Requirements for verifying data base consistency/concurrency	1.8									
(5) Procedures for verifying data base consistency/concurrency			2.8						2.8	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: DISTRIBUTEDNESS	FACTOR(S): SURVIVABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
DI.1 DESIGN STRUCTURE CHECKLIST (a): (1) Design organization identifies all functions and interfaces	1.1		2.1		3.1				2.1	
(2) Data base organization identifies all data and data flow.	1.1		2.1		3.1				2.1	
(3) Specific requirements for information distribution within the data base.	1.1									
(4) Provisions for file/library access from other nodes.	1.8		2.8						2.8	
(5) Provisions for selecting alternate processing capabilities.			2.1						2.1	
(6) Critical system functions distributed over redundant elements/nodes.			2.1						2.1	
(7) Distribution of control functions ensures network operation/integrity under anomalous conditions.			2.1						2.1	
(8) Logical structure and function separated in the design.			2.1							
(9) Physical structure and function separated in the design.			2.1							
(10) Number of nodes that can be removed and still have each node able to communicate with each remaining node (Kleitman's algorithm).			2.1							
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: DOCUMENT ACCESSIBILITY	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
DA.1 ACCESS NO-CONTROL (a): (1) Is there no access control to the software document?	1.11									
METRIC VALUE = 1 (YES) 0.5 (Limited Access) 0 (NO)										

CRITERIA: DOCUMENT ACCESSIBILITY	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
DA.2 WELL-STRUCTURED DOCUMENTATION (a): (1) Clearly and simply written documents	1.11									
(2) Neat and carefully drawn software flow charts with adequate information and explanation	1.11									
(3) Hierarchical structured table of contents used in documents	1.11									
(4) Index system used in documents	1.11									
(5) Separate volumes based on function provided	1.11									
(6) Provide global information about the functional range of the system	1.11									
(7) Describe the functions performed	1.11									
(8) Describe the algorithm used and limitations	1.11									
(9) Describe the relationship between functions	1.11									
(10) Provide software program listing	1.11									
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: DOCUMENT ACCESSIBILITY	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
DA.3 SELECTIVE USABILITY (a): (1) Options available to the user so that selected computation or output feature may be requested	1.11						4.6			
(2) Modules allow for modifying resource utilization i.e., thru use of variable dimensioned arrays							4.6			
(3) Required new functions can be satisfied by using existing design. $\frac{\text{\# functions associated with new application}}{\text{total \# functions performed}}$	1.11									
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: EFFECTIVENESS	FACTOR(S): EFFICIENCY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
EF.1 PERFORMANCE REQUIREMENTS (a): (1) Performance requirements and limitations specified and allocated to functions/design	1.3		2.3		3.3					
METRIC VALUE = 1 (YES) 0 (NO)										

CRITERIA: EFFECTIVENESS	FACTOR(S): EFFICIENCY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
EF.2 ITERATIVE PROCESSING EFFICIENCY MEASURE (by module) (a): (1) Non-loop dependent computations kept out of loop $1 - \frac{\text{\# nonloop dependent statements in loop}}{\text{total \# loop statements}}$						3.3		4.3		
(2) Performance optimizing compiler/assembly language used					3.3					
(3) Compound expressions defined once $1 - \frac{\text{\# compound expression defined more than once}}{\text{\# compound expressions}}$								4.3		
(4) Number of overlays $\frac{1}{\text{\# of overlays}}$				2.3						2.3
(5) Free of bit/byte packing/unpacking in loops			2.3		3.3		4.3			
(6) Module linkages $1 - \frac{\text{module linkage time}}{\text{execution time}}$								4.11		

CRITERIA: EFFECTIVENESS	FACTOR(S): EFFICIENCY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
EF.2 ITERATIVE PROCESSING EFFICIENCY MEASURE (Continued) (7) OS linkages $1 - \frac{\text{OS linkage time}}{\text{execution time}}$								4.11		
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: EFFECTIVENESS	FACTOR(S): EFFICIENCY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
EF.3 DATA USAGE EFFICIENCY MEASURE (by module)(a): (1) Data grouped for efficient processing			2.3						2.3	
(2) Variables initialized when declared $\frac{\text{\# initialized when declared}}{\text{total \# variables}}$								4.3		
(3) No mix-mode expressions $1 - \frac{\text{\# mix mode expressions}}{\text{\# executable statements}}$								4.3		
(4) Common choice of units/type 1/# occurrences of uncommon unit operations								4.9		
(5) Data indexed or referenced for efficient processing			2.3		3.3		(3.3)			
(6) Static data $\frac{\text{\# static data items}}{\text{data base size}}$								4.11		
(7) Dynamic data $\frac{\text{\# modified data items}}{\text{data base size}}$								4.11		
MODULE METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: EFFECTIVENESS	FACTOR(S): EFFICIENCY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
EF.4 STORAGE EFFICIENCY MEASURE (by module) (a)										
(1) Storage requirements allocated to design			2.3							
(2) Virtual storage facilities used			2.3						2.3	
(3) Common data defined only once $1 - \frac{\text{\# variables defined more than once}}{\text{total \# variables}}$								4.9		
(4) Program segmentation $1 - \frac{\text{maximum segment length}}{\text{total program length}}$			2.3					4.11		2.3
(5) Dynamic memory management utilized			2.3						2.3	
(6) Data packing used							4.3			
(7) Storage optimizing compiler/assembly language used			2.3						2.3	
(8) Data base files/libraries stored at only one node.			2.3						2.3	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: FUNCTIONAL OVERLAP	FACTOR(S): INTEROPERABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
FO.1 FUNCTIONAL OVERLAP MEASURE (p): (1) Number of duplicated functions in the systems that are to interoperate $1 - \frac{\text{\# duplicated functions}}{\text{total \# functions}}$		1.13								
(2) Number of duplicated functions to be deleted in one or the other system $1 - \frac{\text{\# deleted functions}}{\text{total \# functions}}$		1.13								
(3) Number of duplicated function pairs to be synchronized $1 - \frac{\text{\# pairs to be synchronized}}{\text{total \# functions}}$		1.13								
(4) Number of duplicated function pairs requiring redundancy management logic to combine them $1 - \frac{\text{\# function pair with redundancy management}}{\text{total \# functions}}$		1.13								
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: FUNCTIONAL SCOPE	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
FS.1 FUNCTION SPECIFICITY (p): (1) Number of functions performed per module $\frac{1}{\text{\# functions performed per module}}$						3.1				
(2) Each module has code comments about functional description $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.6		
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: FUNCTIONAL SCOPE	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
FS.2 FUNCTION COMMONALITY (p): (1) Is the function constructed in a manner which facilitates or encourages its use elsewhere either in part or in total	1.5									
(2) Are the input quantities well defined	1.5									
(3) Are the input formats well defined							4.7			
(4) Are the outputs or database well defined and easy to interpret	1.5									
(5) Does the function performance satisfy one of the specified requirements	1.5									
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: FUNCTIONAL SCOPE	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
FS.3 FUNCTION COMPLETENESS (p): (1) Number of functional requirements satisfied by the reusable software $\frac{\text{\# requirements satisfied}}{\text{total \# requirements}}$		1.5								
METRIC VALUE = Same as entry above										

CRITERIA: GENERALITY	FACTOR(S): FLEXIBILITY, REUSABILITY, EXPANDABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
GE.1 MODULE REFERENCE BY OTHER MODULES (p): (1) Number of modules that are referenced by other modules. $\frac{\text{\# common modules}}{\text{total \# modules}}$				2.6						2.6
METRIC VALUE: Same as entry above										

CRITERIA: GENERALITY	FACTOR(S): FLEXIBILITY, REUSABILITY, EXPANDABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
GE.2 IMPLEMENTATION FOR GENERALITY CHECKLIST (p): (1) Input, processing, output functions are not mixed in a single module.					3.6					
(2) Application and machine-dependent functions are not mixed in a single module. $\frac{1}{\text{\# Machine dependent functions}}$						3.6				
(3) Processing not data volume limited					3.6		4.6			
(4) Processing not data value limited					3.6		4.6			
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: INDEPENDENCE	FACTOR(S): PORTABILITY, REUSABILITY, INTEROPERABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
ID.1 SOFTWARE SYSTEM INDEPENDENCE MEASURE (p): (1) Dependence on software system utility programs, system library routines, and other system facilities $1 - \frac{\text{\# system references}}{\text{Total LOC}}$						3.5		4.5		
(2) Common, standard subset of language used $1 - \frac{\text{\# module violate rule}}{\text{total \# modules}}$						3.5		4.10		
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: INDEPENDENCE	FACTOR(S): PORTABILITY, REUSABILITY, INTEROPERABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
ID.2 MACHINE INDEPENDENCE MEASURE (p): (1) Programming language used available on other machines					3.5		(3.5)			
(2) Free from input/output references $\frac{\text{\# I/O references}}{\text{Total LOC}}$						3.5		4.7		
(3) Code is independent of word and character size $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.10		
(4) Data representation machine independent $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.10		
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: MODULARITY	FACTOR(S): SURVIVABILITY, MAINTAINABILITY, FLEXIBILITY, PORTABILITY, REUSABILITY, INTEROPERABILITY, EXPANDABILITY, VERIFIABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
MO.2 MODULAR IMPLEMENTATION MEASURE (p): (1) Hierarchical structure $1 - \frac{\text{\# modules with violations of hierarchy}}{\text{total \# modules}}$			2.1					4.5		
(2) Module Size Profile If less than 100 locations excluding comments = 1, If not, = 100/Number of locations								4.5		
(3) Controlling parameters defined by calling module $\frac{\text{\# control variables}}{\text{\# calling parameters}}$						3.5		4.5		
(4) Input data controlled by calling module					3.5			4.5		
(5) Output data provided to calling module					3.5			4.5		
(6) Control returned to calling module					3.5			4.5		
(7) Modules do not share temporary storage					3.5		(3.5)			
(8) Each module represents one function $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$				2.1						
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: MODULARITY	FACTOR(S): SURVIVABILITY, MAINTAINABILITY, FLEXIBILITY, PORTABILITY, REUSABILITY, INTEROPERABILITY, EXPANDABILITY, VERIFIABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
MO.3 MODULAR DESIGN MEASURE (p): (1) Processes/functions/modules have loose coupling	1.1		2.1		3.1					
(2) Processes/functions/modules have high cohesion COHESION TYPE — VALUE Functional — 1.0 Informational — 0.7 Communicational — 0.5 Procedural — 0.3 Classical — 0.1 Logical — 0.1 Coincidental — 0.0	1.1			2.1		3.1				
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: OPERABILITY	FACTOR(S): USABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
OP.1 OPERABILITY CHECKLIST (a): (1) All steps of operation described (normal and alternative flows)	1.9		2.9						2.9	
(2) All error conditions and responses appropriately described to operator	1.9		2.9						2.9	
(3) Provisions for operator to interrupt, obtain operational status, save, modify, and continue processing	1.9		2.9						2.9	
(4) Number of operator actions reasonable $1 - \frac{\text{time for operator actions}}{\text{total time for job}}$				2.9						2.9
(5) Job set up and tear down procedures described			2.9						2.9	
(6) Hard copy log of interactions maintained			2.9						2.9	
(7) Operator messages consistent and responses standard			2.9						2.9	
(8) Access violations and responses appropriately described			2.9						2.9	
(9) Capability for operator to obtain network resource status	1.9		2.9						2.9	
(10) Capability to select different nodes for different types of processing or for different types of information retrieval	1.9		2.9						2.9	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: RECONFIGURABILITY	FACTOR(S): SURVIVABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
RE.1 RESTRUCTURE CHECKLIST (a): (1) Configuration of communication links is such that failure of one node/link will not disable communication among other nodes.	1.7		2.7						2.7	
(2) Specific requirements for maintaining data base integrity under anomalous conditions	1.8									
(3) Provisions for maintaining data base integrity under anomalous conditions			2.8						2.8	
(4) Node can rejoin the network when it has been recovered.	1.7		2.7						2.7	
(5) Data replicated at two or more distinct nodes.			2.7						2.7	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SELF-DESCRIPTIVENESS	FACTOR(S): FLEXIBILITY, MAINTAINABILITY, VERIFIABILITY PORTABILITY, REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SD.1 QUANTITY OF COMMENTS (by module) (p): (1) Number of lines of source code and non-blank comments $\frac{\text{\# of comments (nonblank)}}{\text{Total \# lines of source code (nonblank)}}$								4.8		
METRIC VALUE = $\frac{\text{Sum of quantity of comment measures for each module}}{\text{total \# modules}}$										

CRITERIA: SELF-DESCRIPTIVENESS	FACTOR(S): FLEXIBILITY, MAINTAINABILITY, VERIFIABILITY PORTABILITY, REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SD.2 EFFECTIVENESS OF COMMENTS MEASURE (p): (1) Modules have standard formated prologue comments which describe: - Module name/version number - Author - Date - Purpose - Inputs - Outputs - Function - Assumptions - Limitations and restrictions - Accuracy requirements - Error recovery procedures - References $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.8		
(2) Comments set off from code in uniform manner $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.8		

CRITERIA: SELF-DESCRIPTIVENESS	FACTOR(S): FLEXIBILITY, MAINTAINABILITY, VERIFIABILITY PORTABILITY, REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SD.2 EFFECTIVENESS OF COMMENTS MEASURE (CONTINUED): (3) All transfers of control & destinations commented $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.8		
(4) All machine dependent code commented $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.8		
(5) All non-standard HOL statements commented $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.8		
(6) Attributes of all declared variables commented $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.8		
(7) Comments do not just repeat operation described in language $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.8		
(8) Comments about all parameters range values and their default conditions $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.9		
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SELF-DESCRIPTIVENESS	FACTOR(S): FLEXIBILITY, MAINTAINABILITY, VERIFIABILITY PORTABILITY, REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SD.3 DESCRIPTIVENESS OF LANGUAGE MEASURE (p): (1) High order language used $1 - \frac{\text{\# modules with direct code}}{\text{total \# modules}}$								4.8		
(2) Variable names (mnemonics) descriptive of physical or functional property represented $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.8		
(3) Source code logically blocked and indented $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.8		
(4) One statement per line $1 - \frac{\text{\# continuations + multiple statement lines}}{\text{total \# lines}}$								4.8		
(5) Standard format for organization of modules followed $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.8		

CRITERIA: SELF-DESCRIPTIVENESS	FACTOR(S): FLEXIBILITY, MAINTAINABILITY, VERIFIABILITY PORTABILITY, REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SD.3 DESCRIPTIVENESS OF LANGUAGE MEASURE (Continued): (6) No language keywords used as names $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.8		
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SIMPLICITY	FACTOR(S): RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, REUSABILITY, EXPANDABILITY, CORRECTNESS									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SI.1 DESIGN STRUCTURE MEASURE (p): (1) Design organized in hierarchical fashion.			2.1		3.1				2.1	
(2) Module independence	1.1		2.1							
(3) Module processing not dependent on prior processing					3.1					
(4) Each module description includes input, output processing, limitations.					3.1					
(5) Each module has single entrance, single exit. $\frac{1}{2(\#\text{ entrances})} + \frac{1}{(2\ \#\text{ exits})}$						3.1		4.1		
(6) Size of data base				2.8						2.8
(7) Compartmentalization of data base $\frac{\#\text{ files}}{\text{size}}$				2.8						2.8
(8) Programming standard developed			2.1							
(9) Module descriptions include identification of module interfaces.			2.1		3.1					
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\#\text{ applicable elements}}$										

CRITERIA: SIMPLICITY	FACTOR(S): RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, REUSABILITY, EXPANDABILITY, CORRECTNESS									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SI.2 STRUCTURED LANGUAGE OR PREPROCESSOR (a): (1) Structured language or preprocessor used							4.1			
METRIC VALUE: If used = 1, if not used = 0.										

CRITERIA: SIMPLICITY METRIC	FACTOR(S): RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, REUSABILITY, EXPANDABILITY, CORRECTNESS									
	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SI.3 DATA AND CONTROL FLOW COMPLEXITY MEASURE (p): (1) Complexity measure (a) Number of decision points (b) Number of branch points $\frac{1}{1 + \text{\# branchpoints} + \text{\# decision points}}$					3.1		4.1			
METRIC VALUE: $\frac{\text{Sum of complexity measures for each module}}{\text{\# modules}}$										

CRITERIA: SIMPLICITY	FACTOR(S): RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, REUSABILITY, EXPANDABILITY, CORRECTNESS									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SI.4 CODING SIMPLICITY MEASURE (p): (1) Module flow top to bottom.							4.1			
(2) Negative Boolean or complicated compound Boolean expressions used. $1 - \frac{\text{\# of above}}{\text{\# executable statements}}$								4.1		
(3) Jumps in and out of loops $\frac{\text{\# single entry/single exit loops}}{\text{total \# loops}}$								4.1		
(4) Loop index modified $1 - \frac{\text{\# loop indices modified}}{\text{total \# loops}}$								4.1		
(5) Module is not self-modifying $\frac{\text{\# Constructs}}{\text{Total \# LOC}}$								4.1		
(6) Number of statement labels. $1 - \frac{\text{\# labels}}{\text{\# executable statements}}$								4.1		
(7) Nesting level $\frac{1}{\text{max nesting level}}$								4.1		

CRITERIA: SIMPLICITY METRIC	FACTOR(S): RELIABILITY, MAINTAINABILITY, VERIFIABILITY, FLEXIBILITY, REUSABILITY, EXPANDABILITY, CORRECTNESS									
	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SI.4 CODING SIMPLICITY MEASURE (Continued): (8) Number of branches $1 - \frac{\text{\# branches}}{\text{\# executable statements}}$								4.1		
(9) Statement simplicity level $1 - \frac{\text{\# declarative + \# data manipulation statements}}{\text{\# executable statements}}$								4.1		
(10) Variable mix in a module $\frac{\text{\# internal variables}}{\text{total \# variables}}$								4.9		
(11) Variable density $1 - \frac{\text{\# variables}}{\text{\# exec statements}}$								4.9		
(12) Single use of variables.							4.9			
(13) Code written according to a programming standard.	1.1				3.1		4.1			
(14) Macros and subroutines used to avoid repeated and redundant code.					3.1		4.1			
METRIC VALUE: $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SPECIFICITY	FACTOR(S): CORRECTNESS, VERIFIABILITY, EXPANDABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SP.1 SCOPE OF FUNCTION MEASURE (p): (1) Input density $\frac{1}{1 + \#\text{ input parameters}}$						3.1				
(2) Output density $\frac{\#\text{ output parameters}}{\#\text{ output values used}}$						3.1				
(3) Same function cannot be accomplished by multiple variant forms.					3.1					
METRIC VALUE: $\frac{\text{Total score from applicable elements}}{\#\text{ applicable elements}}$										

CRITERIA: SYSTEM ACCESSIBILITY	FACTOR(S): INTEGRITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SA.1 ACCESS CONTROL CHECKLIST (a): (1) User I/O access controls provided (ID's, passwords)	1.12		2.12						2.12	
(2) Data base access controls provided (authorization tables, privacy locks)	1.12		2.12						2.12	
(3) Memory protection across tasks provided	1.12		2.12						2.12	
(4) Network access controls provided	1.12		2.12						2.12	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SYSTEM ACCESSIBILITY	FACTOR(S): INTEGRITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SA.2 ACCESS AUDIT CHECKLIST (a): (1) Provisions for recording and reporting access to system	1.12		2.12						2.12	
(2) Provisions for immediate indication of access violations	1.12		2.12						2.12	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SYSTEM CLARITY	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SC.1 INTERFACE COMPLEXITY (p): (1) Number of data items (variable names) used to specify the interface $\frac{1}{1 + \text{\# data items specified interface}}$								4.1		
(2) Number of data items passed implicitly across interface via common global data without adequate comments $\frac{1}{1 + \text{\# data items passed implicitly across interface without comments}}$								4.1		
(3) Number of nesting levels in interface $\frac{1}{\text{\# nesting levels in interface}}$								4.1		
(4) Number of interface data items with negative qualification $1 - \frac{\text{\# data items with negative qualification}}{\text{total \# data items}}$								4.1		
(5) Number of data items passed across module interface via module arguments and values or via common global data $\frac{1}{1 + \text{\# data items passed across module interface}}$								4.1		

CRITERIA: SYSTEM CLARITY	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SC.1 INTERFACE COMPLEXITY (Continued): (6) Module interfaces-established by common control blocks or common data blocks or common overlay region of memory or common I/O devices or global variable names and without adequate comments (= 0 yes; = 1 no)							4.1			
(7) Modules do not modify other modules $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.1		
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SYSTEM CLARITY	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SC.2 PROGRAM FLOW COMPLEXITY (p): (1) Number of possible unique execution paths $\frac{1}{\text{\# unique paths}}$								4.1		
(2) Number of IF statements $\frac{1}{1 + \text{\# IF statements in each module}}$								4.1		
(3) Number of function CALLs in each module $\frac{1}{1 + \text{\# CALLs in each module}}$								4.1		
(4) Number of control variables used to direct execution path selection $\frac{1}{1 + \text{\# control variables}}$								4.1		
(5) Number of DO groups $\frac{1}{1 + \text{\# DO groups in each module}}$								4.1		
(6) Each module has code comments that indicate called-by modules and calling modules $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.1		
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SYSTEM CLARITY	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SC.3 APPLICATION FUNCTIONAL COMPLEXITY (p): (1) Separate input/output from computational functions $1 - \frac{\text{\# modules with mixed I/O \& computational}}{\text{total \# modules}}$				2.1						2.1
(2) Modules do not share temporary storage locations $1 - \frac{\text{\# modules with sharing temporary storage}}{\text{total \# modules}}$								4.1		
(3) Separate database - management routines and storage-management routines $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.1		
(4) Common function is not distributed among different modules			2.1						2.1	
(5) Module is not made to do too many (related but differe.) functions			2.1						2.1	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SYSTEM CLARITY	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SC.4 COMMUNICATION COMPLEXITY (p): (1) Number of formal parameters each routine $\frac{\text{\# parameters each routine}}{\text{\# of global variables}}$								4.1		
(2) Common global variable used each module $\frac{1}{1 + \text{\# common global variables in the module}}$								4.1		
(3) Routine-Global-Routine data binding $\frac{\text{\# global variables which are modified by first routine \& referenced by second routine}}{\text{total \# global variables}}$								4.1		
(4) Modules connections are established by referring to other modules by their functional names, not internal elements of other modules $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.1		
(5) Communication between modules is by passing data, not by passing control elements $1 - \frac{\text{\# modules violate rule}}{\text{total \# modules}}$								4.1		
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SYSTEM CLARITY	FACTOR(S): REUSABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SC.5 STRUCTURE CLARITY (p): (1) Do not compute the same value more than once							4.1			
(2) Do not insert a statement which never needs to be executed							4.1			
(3) Maintain a constant meaning for each variable							4.1			
(4) Eliminate unnecessary intermediate variables							4.1			
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SYSTEM COMPATIBILITY	FACTOR(S): INTEROPERABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SY.1 COMMUNICATION COMPATIBILITY CHECKLIST (a): (1) Same I/O transmission rates in both systems?			2.11						2.11	
(2) Same communication protocol in both systems?			2.11						2.11	
(3) Same message content in both systems?			2.11						2.11	
(4) Same message structure and sequence in both systems?			2.11						2.11	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SYSTEM COMPATIBILITY	FACTOR(S): INTEROPERABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SY.2 DATA COMPATIBILITY CHECKLIST (a): (1) Is data in both systems in the same format (ASCII, EBCDIC, ...)?			2.11						2.11	
(2) Same data base structure in both systems?			2.11						2.11	
(3) Same data base access techniques in both systems?			2.11						2.11	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SYSTEM COMPATIBILITY	FACTOR(S): INTEROPERABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SY.3 HARDWARE COMPATIBILITY CHECKLIST (a): (1) Same word length in both systems?			2.11							
(2) Same interrupt structure in both systems?			2.11							
(3) Same instruction set in both systems?			2.11							
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SYSTEM COMPATIBILITY	FACTOR(S): INTEROPERABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SY.4 SOFTWARE COMPATIBILITY CHECKLIST (a): (1) Same source language in both systems?			2.11							
(2) Same operating system in both systems?			2.11							
(3) Same support software in both systems?			2.11							
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: SYSTEM COMPATIBILITY	FACTOR(S): INTEROPERABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
SY.5 DOCUMENTATION FOR OTHER SYSTEMS (a): (1) Is the other system documentation available in a form that is up-to-date, complete, and clearly organized and written?	1.11									
METRIC VALUE = 1 (YES) 0 (NO)										

CRITERIA: TRACEABILITY	FACTOR(S): CORRECTNESS									
METRIC	Requirements		Prel Design		Detail Design		Implemen-tation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
TR.1 CROSS REFERENCE (a): (1) Cross reference relating functions/modules to requirements	1.4		2.4		3.4					
METRIC VALUE = 1 (YES) 0 (NO)										

CRITERIA: TRAINING	FACTOR(S): USABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
TN.1 TRAINING CHECKLIST (a): (1) Lesson plans/training material developed for operators, end users, maintainers			2.9						2.9	
(2) Realistic simulated exercises provided			2.9						2.9	
(3) Sufficient 'help' and diagnostic information available on-line			2.9						2.9	
(4) Selectable levels of aid and guidance for users of different degrees of expertise.			2.9						2.9	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

CRITERIA: VIRTUALITY	FACTOR(S): INTEGRITY, USABILITY, EXPANDABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
VR.1 SYSTEM/DATA INDEPENDENCE CHECKLIST (a): (1) Specific requirements for virtual storage structure	1.8									
(2) Provisions for virtual storage structure (User can obtain data without knowing identity/location of storage device).			2.8						2.8	
(3) Users can manipulate data as if it were not replicated elsewhere in the system.			2.8						2.8	
(4) Each user can utilize system as though it were dedicated to that user.			2.1						2.1	
(5) User is presented with a complete logical system without regard to physical topology.			2.1						2.1	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

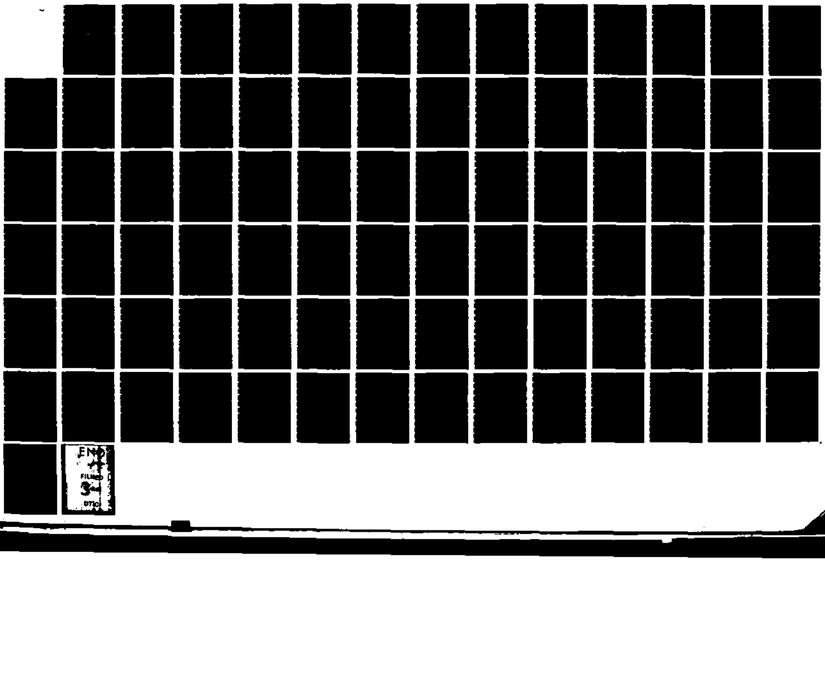
CRITERIA: VISIBILITY	FACTOR(S): VERIFIABILITY, USABILITY, MAINTAINABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
VS.1 MODULE TESTING MEASURE (by module) (a): (1) Path coverage $\frac{\text{\# paths to be tested}}{\text{total \# paths}}$				2.10						2.10
(2) Input parameters boundary tested $\frac{\text{\# parameters to be boundary tested}}{\text{total \# parameters}}$				2.10						2.10
METRIC VALUE = $\frac{\text{Sum of module testing measures for each module}}{\text{total \# modules}}$										

CRITERIA: VISIBILITY METRIC	FACTOR(S): VERIFIABILITY, USABILITY, MAINTAINABILITY									
	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
VS.2 INTEGRATION TESTING MEASURE (a): (1) Module interfaces tested $\frac{\text{\# to be tested}}{\text{total \# interfaces}}$				2.10						2.10
(2) Performance requirements (timing & storage) coverage $\frac{\text{\# requirements to be tested}}{\text{total \# performance requirements}}$				2.10						2.10
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

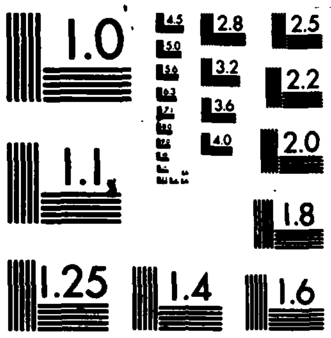
CRITERIA: VISIBILITY	FACTOR(S): VERIFIABILITY, USABILITY, MAINTAINABILITY									
METRIC	Requirements		Prel Design		Detail Design		Implementation		Test & Integration	
	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value	Yes/No 1 or 0	Value
VS.3 SYSTEM TESTING MEASURE (a): (1) Module coverage (for all test scenarios) $\frac{\text{\# modules to be executed}}{\text{total \# of modules}}$				2.10						2.10
(2) Identification of test inputs and outputs in summary form			2.10						2.10	
METRIC VALUE = $\frac{\text{Total score from applicable elements}}{\text{\# applicable elements}}$										

AD-A137 956 SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS VOLUME 2 GUIDEBOOK F..(U) BOEING AEROSPACE CO SEATTLE WA T P BOWEN ET AL. JUL 83 RADC-TR-83-175-VOL-2 3/3

UNCLASSIFIED F30602-80-C-0330 F/G 9/2 NL



END
FILMED
3-84
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

APPENDIX C
METRIC EXPLANATIONS

Appendix C contains a detailed explanation of each metric element. The explanations are organized alphabetically by quality criteria and numerically by metric acronym. A summary of the metric explanations is shown on the next several pages. For each metric element, the definition (from Appendix B, Metric Tables) is stated, and an explanation of the element is provided.

The contents of this appendix are based on the results of this contract, "Quality Metrics for Distributed Systems", F30602-80-C-0330 and the results of contract F30602-80-C-0265, "Software Interoperability and Reusability". This appendix includes a refinement and reorganization of metric explanation information initially defined in RADC-TR-77-369 and RADC-TR-80-109.

METRIC EXPLANATION SUMMARY

CRITERIA	ACRONYM	METRICS
ACCURACY	AY.1	ACCURACY CHECKLIST
ANOMALY MANAGEMENT	AM.1	ERROR TOLERANCE/CONTROL CHECKLIST
	AM.2	IMPROPER INPUT DATA CHECKLIST
	AM.3	COMPUTATIONAL FAILURES CHECKLIST
	AM.4	HARDWARE FAULTS CHECKLIST
	AM.5	DEVICE ERRORS CHECKLIST
	AM.6	COMMUNICATION ERRORS CHECKLIST
	AM.7	NODE/COMMUNICATIONS FAILURES CHECKLIST
APPLICATION INDEPENDENCE	AI.1	DATA BASE SYSTEM INDEPENDENCE
	AI.2	DATA STRUCTURE
	AI.3	ARCHITECTURE STANDARDIZATION
	AI.4	MICROCODE INDEPENDENCE
	AI.5	ALGORITHM
AUGMENTABILITY	AG.1	DATA STORAGE EXPANSION MEASURE
	AG.2	COMPUTATION EXTENSIBILITY MEASURE
	AG.3	CHANNEL EXTENSIBILITY MEASURE
	AG.4	DESIGN EXTENSIBILITY CHECKLIST
AUTONOMY	AU.1	INTERFACE COMPLEXITY MEASURE
	AU.2	SELF-SUFFICIENCY CHECKLIST
COMMONALITY	CL.1	COMMUNICATIONS COMMONALITY CHECKLIST
	CL.2	DATA COMMONALITY CHECKLIST
	CL.3	COMMON VOCABULARY CHECKLIST
COMMUNICATIVENESS	CM.1	USER INPUT INTERFACE MEASURE
	CM.2	USER OUTPUT INTERFACE MEASURE
COMPLETENESS	CP.1	COMPLETENESS CHECKLIST
CONCISENESS	CO.1	HALSTEAD'S MEASURE
CONSISTENCY	CS.1	PROCEDURE CONSISTENCY MEASURE
	CS.2	DATA CONSISTENCY MEASURE
DISTRIBUTEDNESS	DI.1	DESIGN STRUCTURE CHECKLIST
DOCUMENT ACCESSIBILITY	DA.1	ACCESS NO-CONTROL
	DA.2	WELL-STRUCTURED DOCUMENTATION
	DA.3	SELECTIVE USABILITY

METRIC EXPLANATION SUMMARY

CRITERIA	ACRONYM	METRICS
EFFECTIVENESS	EF.1	PERFORMANCE REQUIREMENTS
	EF.2	ITERATIVE PROCESSING EFFICIENCY MEASURE
	EF.3	DATA USAGE EFFICIENCY MEASURE
	EF.4	STORAGE EFFICIENCY MEASURE
FUNCTIONAL OVERLAP	FO.1	FUNCTIONAL OVERLAP MEASURE
FUNCTIONAL SCOPE	FS.1	FUNCTION SPECIFICITY
	FS.2	FUNCTION COMMONALITY
	FS.3	FUNCTION COMPLETENESS
GENERALITY	GE.1	MODULE REFERENCE BY OTHER MODULES
	GE.2	IMPLEMENTATION FOR GENERALITY CHECKLIST
INDEPENDENCE	ID.1	SOFTWARE SYSTEM INDEPENDENCE MEASURE
	ID.2	MACHINE INDEPENDENCE MEASURE
MODULARITY	MO.2	MODULAR IMPLEMENTATION MEASURE
	MO.3	MODULAR DESIGN MEASURE
OPERABILITY	OP.1	OPERABILITY CHECKLIST
RECONFIGURABILITY	RE.1	RESTRUCTURE CHECKLIST
SELF-DESCRIPTIVENESS	SD.1	QUANTITY OF COMMENTS
	SD.2	EFFECTIVENESS OF COMMENTS MEASURE
	SD.3	DESCRIPTIVENESS OF LANGUAGE MEASURE
SIMPLICITY	SI.1	DESIGN STRUCTURE MEASURE
	SI.2	STRUCTURED LANGUAGE OR PRE-PROCESSOR
	SI.3	DATA AND CONTROL FLOW COMPLEXITY MEASURE
	SI.4	CODING SIMPLICITY MEASURE
SPECIFICITY	SP.1	SCOPE OF FUNCTION MEASURE
SYSTEM ACCESSIBILITY	SA.1	ACCESS CONTROL CHECKLIST
	SA.2	ACCESS AUDIT CHECKLIST

METRIC EXPLANATION SUMMARY

CRITERIA	ACRONYM	METRICS
SYSTEM CLARITY	SC.1	INTERFACE COMPLEXITY
	SC.2	PROGRAM FLOW COMPLEXITY
	SC.3	APPLICATION FUNCTIONAL COMPLEXITY
	SC.4	COMMUNICATION COMPLEXITY
	SC.5	STRUCTURE CLARITY
SYSTEM COMPATIBILITY	SY.1	COMMUNICATION COMPATIBILITY CHECKLIST
	SY.2	DATA COMPATIBILITY CHECKLIST
	SY.3	HARDWARE COMPATIBILITY CHECKLIST
	SY.4	SOFTWARE COMPATIBILITY CHECKLIST
	SY.5	DOCUMENTATION FOR OTHER SYSTEM
TRACEABILITY	TR.1	CROSS REFERENCE
TRAINING	TN.1	TRAINING CHECKLIST
VIRTUALITY	VR.1	SYSTEM/DATA INDEPENDENCE CHECK-LIST
VISIBILITY	VS.1	MODULE TESTING MEASURE
	VS.2	INTEGRATION TESTING MEASURE
	VS.3	SYSTEM TESTING MEASURE

Criteria: <u>Accuracy</u>

Metric: AY.1 Accuracy Checklist.

Each element is a binary measure indicating existence or absence of the elements. The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Error analysis performed and budgeted to module.

An error analysis must be part of the requirements analysis performed to develop the requirements specification. This analysis allocates overall accuracy requirements to the individual functions to be performed by the system. This budgeting of accuracy requirements provides definitive objectives to the module designers and implementers.

(2) A definitive statement of requirement for accuracy of inputs, outputs, processing, and constants.

See explanation (1) above.

(3) Sufficiency of math library.

The accuracy of the math library routines utilized within the system is to be checked for consistency with the overall accuracy objectives.

(4) Sufficiency of numerical methods.

The numerical methods utilized within the system are to be consistent with the accuracy objectives.

(5) Execution outputs within tolerances.

A final measure during development testing is execution of modules and checking for accuracy of outputs.

(6) Accuracy requirements budgeted to functions/modules.

The budgeting of accuracy requirements is repeated at succeedingly lower levels of design - during preliminary and detail design.

Criteria: <u>Anomaly Management</u>

Metric: AM.1 Error Tolerance/Control Checklist.

The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Concurrent processing centrally controlled.
Functions which may be used concurrently are to be controlled centrally to provide concurrency checking, read/write locks, etc. Examples are a data base manager, I/O handling, error handling, etc.

(2) Errors fixable and processing continued.
When an error is detected, the capability to correct it on-line and then continue processing should be available. An example is an operator message that the wrong tape is mounted and processing will continue when correct tape is mounted.

(3) When an error condition is detected, the condition is to be passed up to calling routine.
The decision of what to do about an error is to be made at a level where an affected module is controlled. This concept is built into the design and then implemented.

(4) Any parallel processing centrally controlled.
When parallel processing is performed it is controlled by concurrent inputs, by concurrent output checks, and/or by comparing output results.

Criteria: Anomaly Management

Metric: AM.2 Improper Input Data checklist.
The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) A definitive statement of requirement for error tolerance of input data.
The requirements specification must identify the error tolerance capabilities desired.

(2) Range of values (reasonableness) for items specified and checked.
The attribute of each input item is to be checked for reasonableness. Examples are checking items if they must be numeric, alphabetic, positive or negative, of a certain length, nonzero, etc. These checks are to be specified at design and exist in code at implementation.

(3) Conflicting requests and illegal combinations identified and checked.
Checks to see if redundant input data agrees, if combinations of parameters are reasonable, and if requests are conflicting. These checks should be documented in the design and exist in the code at implementation.

(4) All input is checked before processing begins.
Input checking is not to stop at the first error encountered but. to continue through all the input and then report all errors. Processing is not to start until the errors are reported and either corrections are made or a continue processing command is given.

(5) Determination that all data is available prior to processing.
To avoid going through several processing steps before incomplete input data is discovered, checks for sufficiency of input data are to be made prior to the start of processing.

Criteria: Anomaly Management

Metric: AM.3 Computational Failures Checklist.

The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) A definitive statement of requirement for recovery from computational failures.
The requirement for this type of error tolerance capability are to be stated during requirements phase.

(2) Loop and multiple transfer index parameters range tested before use.
Range tests for loop indices and multiple transfers are to be specified at design and to exist in code at implementation.

(3) Subscript checking.
Checks for legal subscript values are to be specified at design and coded during implementation.

(4) Critical output parameters reasonableness checked during processing.
Certain range-of-value checks are to be made during processing to ensure the reasonableness of final outputs. This is usually done only for critical parameters. These are to be identified during design and coded during implementation.

Metric: AM.4 Hardware Faults Checklist.

The metric is the sum of scores from the applicable elements divided by the number of applicable elements.

Criteria: <u>Anomaly Management</u>

(1) A definitive statement of requirements for recovery from hardware faults.
The handling of hardware faults such as arithmetic faults, power failure, clock interrupt, etc., are to be specified during the requirements phase.

(2) Recovery from hardware faults.
The design specification and code to provide the recovery from the hardware faults identified in the requirements must exist in the design and implementation phases respectively.

Metric: AM.5 Device Errors Checklist.
The metric is the sum of the scores given to the following applicable elements divided by the number of applicable elements.

(1) A definitive statement of requirements for recovery from device errors.
The handling of device errors such as unexpected end-of-files or end-of-tape conditions and read/write failures are specified during the requirements phase.

(2) Recovery from device errors.
The design specification and code to provide the required handling of device errors must exist in the design and implementation phases respectively.

Metric: AM.6 Communications Errors Checklist.
The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) A definitive statement of requirements for recovery from communication errors.
Explicit requirements are to be stated for recovery from communication errors.

Criteria: Anomaly Management

(2) Provisions for recovery from communication errors.
The preliminary design should reflect a design solution to the stated requirements.

(3) Check sums computed and transmitted with all messages.
Check sums are a common form of detecting communication errors.

(4) Check sums computed and compared upon message reception.
Check sums are a common form of detecting communication errors.

Metric: AM.7 Node/Communications Failures Checklist.
The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) A definitive statement of requirements for recovery from node/communication failures.
Explicit requirements are to be stated for recovery from node/communication failures.

(2) Provisions for recovery from node/communication failures.
The preliminary design should reflect a design solution to the stated requirements.

(3) Adjacent nodes checked for operational status.
Checking adjacent nodes is a common form of detecting node failures.

(4) Alternate strategies for message routing.
Employing an alternate message routing strategy is a common way of recovering from node/communication failures.

Criteria: Application Independence

Metric: AI.1 Database System Independence.

Software which is free from database system reference has higher reusability.

The metric measure is based on how the module is independent of the database system.

(1) Free from database system reference.

The metric is based on the database system reference within a module.

Metric: AI.2 Data Structure.

Generalized data structures which are easy to understand, flexible, and extensible reduce the costs associated with reusing the software. The software with control of data structure has enhanced modifiability, and it tends to be more reusable. The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Data in parameter list, data structure described parametrically.

Parametric definitions of data structures will reduce the reuse software costs. The metric is based on how many data items could be parametrized and parametrized data items.

(2) Data communicated through common storage region and with adequate comments.

To reduce the software reuse costs the data should be centrally controlled such as through global storage. Then common data in a module must have adequate explanations. This is a binary measure.

(3) Control of database structures, both global and local, i.e., all data centrally controlled and symbolically defined and referenced.

See explanation for (2) above.

Criteria: Application Independence

(4) Logical processing independent of data storage specification and requirement.
The software with logical processing independent of data storage will tend to be more reusable. The measure is based on the number of modules which do not comply.

(5) Each module has code comments about data items description including global & parameter input/output and local variables.
See explanation for (2) above.

Metric: AI.3 Architecture Standardization.
Standardization of computer architecture can increase the potential reuse of software by increasing the number of environments in which the software can be executed without change. The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Module is free from computer architecture reference.
When software is independent from computer architecture reference it tends to be more reusable. This is a binary measure.

(2) Module is in standard 32 bits computer architecture (Nebula).
When software is in a standard computer architecture then it will be easier to reuse in another computer with standard architecture. This is a binary measure.

(3) Code statements are free from machine architecture.
See explanation for (1) above.

Criteria: <u>Application Independence</u>

Metric: AI.4 Microcode Independence.

Using the microcode or machine language code in software will reduce the number of environments where software can be reused and also reduce the software flexibility. The metric measure is based on how the module is free from microcode instructions.

(1) Number of modules used microcode instruction.
The metric is based on the microcode references within a module.

Metric: AI.5 Algorithm.

An algorithm that functions well over a wide range of inputs will generally require less modification before it can be reused. The use of table driven algorithms will produce highly reusable software which can be easily adapted to different applications. The metric is the sum of the scores of the following applicable elements divided by the total number of applicable elements.

(1) Valid range.
The range of inputs the function algorithm can handle. The metric is based on the number of the domains the algorithm works for.

(2) Is the algorithm table driven?
The table-driven algorithm can be easily adapted to different applications. The metric is a binary measure.

(3) Is the algorithm certification available?
The software with algorithm certification available tends to be more reusable. The metric is a binary measure.

(4) Is the algorithm test data available?
See explanation for (3) above.

Criteria: <u>Application Independence</u>

(5) Each module has code comments about algorithm description.
The algorithm usage should be explained in the code comments. The measure is based on the number of modules which do not follow this practice.

Criteria: <u>Augmentability</u>

Metric: AG.1 Data Storage Expansion Measure.
The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Logical processing independent of storage specification/requirements.
The logical processing of a module is to be independent of storage size, buffer space, or array sizes. The design provides for variable dimensions and dynamic array sizes to be defined parametrically. The metric is based on the number of modules containing hard-coded dimensions which do not exemplify this concept.

(2) Percent of memory capacity uncommitted.
The amount of memory available for expansion is an important measure. This measure identifies the percent of available memory which has not been utilized in implementing the current system.

(3) Percent auxilliary storage capacity uncommitted.
See explanation for (2) above.

Metric: AG.2 Computation Extensibility Measure.
The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Accuracy, convergence, timing attributes which control processing are parametric.
A module which can provide varying degrees of convergence or timing to achieve greater precision provides this attribute of extensibility. Hard-coded control parameters, counters, clock values, etc. violate this measure. This measure is based on the number of modules which do not exemplify this characteristic.

Criteria: Augmentability

(2) Modules table driven.

The use of tables within a module facilitates different representations and processing characteristics. This measure which can be applied during design and implementation is based on the number of modules which are not table driven.

(3) Percent of speed capacity uncommitted.

A certain function may be required in the performance requirements specification to be accomplished in a specified time for overall timing objectives. The amount of time not used by the current implementation of the function is processing time available for potential expansion of computational capabilities. This measure identifies the percent of total processing time that is uncommitted.

Metric: AG.3 Channel Extensibility Measure.

The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Spare I/O channel capacity (by peripheral).

A load will be placed on the channels to each peripheral because of the design solution. The amount of channel capacity which is uncommitted is the amount available for potential expansion.

(2) Spare communication channel capacity.

A load will be placed on each communication channel because of the design solution. The amount of communication channel capacity which is uncommitted is the amount available for potential expansion.

Metric: AG.4 Design Extensibility Checklist.

The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

Criteria: <u>Augmentability</u>

(1) Processors, communication links, memory devices, and peripherals compatible (of a common vendor or model).
It is desirable to have network hardware compatible as this minimizes interface complexity and eases the task of expansion.

(2) Documentation reveals performance price of software/system for enhancement trades.
The cost required to achieve the specified performance levels has seldom been documented; yet this is an essential element in performing trades for enhancing the system.

(3) Specifications identify new technology tradeoff areas for software.
This information would be useful for future changes in the software and the system.

(4) Software specifications include requirements for the criteria of the quality factor expandability.
Building in the expansion capability will minimize future costs.

Criteria: Autonomy

Metric: AU.1 Interface Complexity Measure.
The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Processes/functions separated as logical "wholes" to minimize interface complexity.
Minimizing interface complexity in the functional design will aid in keeping interfaces simple in the detail design.

(2) Interface code.
The greater the amount of interface code, in general, the more complex is the interface. This measure identifies the fraction of non-interface code.

(3) Interface modules.
The greater number of interface modules, in general, the more complex is the interface. This measure identifies the fraction of non-interface modules.

(4) Communication loading.
The complexity of the interface is reflected in part by the percentage of use.
This measure identifies the fraction of idle interface communication time.

Metric: AU.2 Self-sufficiency Checklist.
The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Software volatility - each CPU/system has separate power supply.
System software vulnerability is reduced by increasing the independence of each CPU/system.

Criteria: Autonomy

(2) Each scheduling unit (i.e., executive, operating system) tests its own operation, communication links, memories, and peripherals.
System software vulnerability is reduced through independent node self-test.

(3) Software system includes word-processing capability.
System autonomy is enhanced by being able to produce documentation on-site.

Criteria: Commonality

Metric: CL.1 Communications Commonality Checklist.

The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Definitive statement of requirements for communication with other systems.

During the requirement phase, the communication requirements with other systems must be considered. This is a binary measure of the existence of this consideration.

(2) Protocol standards established and followed for network process control.

The communication protocol standards for communication with other systems are to be established during the design phase and followed during implementation. This binary measure applied at each of these phases indicates whether the standards were established and followed.

(3) Single module interface for input (from another system).

The more modules which handle input the more difficult it is to interface with another system and implement standard protocols. This measure is based on the reciprocal of the number of modules which handle input.

(4) Single module interface for output (to another system).

For similar reasons as (3) above this measure is the reciprocal of the number of output modules.

(5) Specific requirements for network process control.

Network process control requirements should be specified during the requirements analysis phase and consider all nodes in the network.

Criteria: Commonality

(6) Specific requirements for user session control.
Requirements for the control of a user session on the network should be specified during the requirements analysis phase and consider all nodes in the network.

(7) Specific requirements for communication routing strategy.
Requirements for communication routing should be specified during the requirements analysis phase and consider all nodes in the network configuration.

(8) Protocol standards established and followed for user session control.
The design and implementation should comply with network-wide protocol standards.

(9) Protocol standards established and followed for communication routing.
The design and implementation should comply with network-wide protocol standards.

(10) Number of systems responding correctly to successfully complete handshaking. The larger the number of systems which must respond correctly, the greater the effort required.

(11) Low time dependency on handshaking. High time dependencies impose greater constraints on computation and response times, which will increase the total effort.

(12) No communication time dependency.
If the communication function has time dependencies, such as freshness of data or response to input data within certain time limits, then the effort increases.

Criteria: Commonality

(13) Number of other systems this system will interface with.
The number of systems with which this system must interoperate should greatly affect the total interoperability effort.

(14) No timing dependency on data freshness.
The requirement for data freshness will increase effort to meet timing factors.

(15) Operating procedures known.
The operating procedures used with the system must be known so the requirements can be understood in context.

Metric: CL.2 Data Commonality Checklist.
The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Definitive statement for standard data representation for communication with other systems.
This is a binary measure of the existence of consideration for standard data representation between systems which are to be interfaced. This must be addressed and measured in the requirements phase.

(2) Translation standards among representations established and followed.
More than one translation from the standard data representations used for interfacing with other systems may exist within a system. Standards for these translations are to be established and followed. This binary measure identifies if the standards are established during design and followed during implementation.

(3) Single module to perform each translation.
This measure is the reciprocal of the maximum number of modules which perform a translation.

Criteria: Commonality

Metric: CL.3 Common Vocabulary Checklist.

The binary metric is the single value answer to the question of common vocabulary use among interoperating systems. If there is more than one system with which the subject system is to interoperate, then the value of this metric is the average of the individual metrics for each interoperating system.

(1) Do both projects use the same technical vocabulary with identical meanings? According to published material on interoperability, one of the most prevalent and pervasive problems is the use of inconsistent terminologies. Projects may use different vocabularies with the same meanings, or use the same vocabulary with different meanings. As a result, people either don't understand each other and know it, or don't understand each other and don't know it. Either way, interoperability problems are the sure result.

Criteria: Communicativeness

Metric: CM.1 User Input Interface Measure.

The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Default values defined.

A method of minimizing the amount of input required is to provide defaults. This measure, applied during design and implementation, is based on the number of defaults allowed divided by the total number of input parameters.

(2) Input format uniform.

The greater the number of input formats there are the more difficult the system is to use. This measure is based on the total number of input formats.

(3) Each input record self-identifying.

Input records which have self-identifying codes enhance the accuracy of user inputs. This measure is based on the number of input records that are not self identifying divided by the total number of input records.

(4) Input can be verified by user prior to execution.

The capability, displaying input upon request or echoing the input automatically, enables the user to check his inputs before processing. This is a binary measure of the existence of the design and implementation of this capability.

(5) Input terminated by explicitly defined logical end of input.

The user should not have to provide a count of input cards. This is a binary measure of the design and implementation of this capability.

Criteria: Communicativeness

(6) Provision for specifying input from different media.
The flexibility of input must be decided during the requirements analysis phase and followed through during design and implementation. This is a binary measure of the existence of the consideration of this capability during all three of these phases.

Metric: CM.2 User Output Interface Measure.
The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Selective output controls.
The existence of a requirement for, design for, and implementation of selective output controls is indicated by this binary measure. Selective controls include choosing specific outputs, output formats, amount of output, etc.

(2) Outputs have unique descriptive user oriented labels.
This is a binary measure of the design and implementation of unique output labels. In addition, the labels are to be descriptive to the user. This includes not only the labels which are used to reference an output report but also the title, column headings, etc. within that report.

(3) Outputs have user oriented units.
This is a binary measure which extends (2) above to the individual output items.

(4) Uniform output labels.
This measure corresponds to (2) above and is the reciprocal of the number of different output formats.

Criteria: Communicativeness

(5) Logical groups of output separated for user examination.
Utilization of top of page, blank lines, lines of asterisks, etc., provide for easy identification of logically grouped output. This binary measure identifies if these techniques are used during design and implementation.

(6) Relationship between error messages and outputs is unambiguous.
This is a binary measure applied during design and implementation which identifies if error messages will be directly related to the output.

(7) Provision for redirecting output to different media.
This is a binary metric which identifies if consideration is given to the capability to redirect output to different media during requirements analysis, design, and implementation.

(8) Standard user interfaces for network information and data access.
This is a binary metric which considers a common user language for accessing information/data throughout the network. This capability relieves the user of the need to know the languages of different nodes.

Criteria: <u>Completeness</u>

Metric: CP.1 Completeness Checklist.

This metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Unambiguous requirements/references for input, function, and output.
Unique references to data or functions avoid ambiguities such as a function being called one name by one module and by another name by another module. Unique references avoid this type of ambiguity in all three phases.

(2) All data references defined, computed, or obtained from an external source.
Each data element is to have a specific origin. At the requirements level only major global data elements and a few specific local data elements may be available to be checked. The set of data elements available for completeness checking at the design level increases substantially and is to be complete at implementation.

(3) All defined functions used.
A function which is defined but not used during a phase is either nonfunctional or a reference to it has been omitted.

(4) All referenced functions defined.
A system is not complete at any phase if dummy functions are present or if functions have been referenced but not defined.

(5) All conditions and processing defined for each decision point.
Each decision point is to have all of its conditions and alternative processing paths defined at each phase of the software development.

Criteria: <u>Completeness</u>

The level of detail to which the conditions and alternative processing are described may vary but the important element is that all alternatives are described.

(6) All defined and referenced calling sequence parameters agree.
For each interaction between modules, the full complement of defined parameters for the interface is to be used. A particular call to a module should not pass, for example, only five of the six defined parameters for that module.

(7) All problem reports resolved.
At each phase in the development, problem reports are generated. Each is to be closed or a reolution indicated to ensure a complete product.

Criteria: Conciseness

Metric: CO.1 Halstead's Measure.

The metric is based on Halstead's concept of length (HALSM77).

The observed length of a module is

$N_0 = N_1 + N_2$ where:

N_1 = total usage of all operands in a module

N_2 = total usage of all operands in a module

The calculated length of a module is

$N_c = n_1 \log_2 n_1 + n_2 \log_2 n_2$ where:

n_1 = number of unique operators in a module

n_2 = number of unique operators in a module

The metric is normalized as follows:

$$1 - \frac{N_c - N_0}{N_0} \qquad \text{or,}$$

$$0 \text{ if } \frac{N_c - N_0}{N_0} \qquad \text{greater than } 1$$

At a system level the metric is the averaged value of all the module metric values.

Criteria: Consistency

Metric: CS.1 Procedure Consistency Measure.

The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Standard design representation.

Flow charts, HIPO charts, Program Design Language - whichever form of design representation is used, standards for representing the elements of control flow are to be established and followed. This element applies to design only. The measure is based on the number of modules whose design representation does not comply with the standards.

(2) Calling sequence conventions.

Interactions between modules are to be standardized. The standards are to be established during design and followed during implementation. The measure is based on the number of modules which do not comply with the conventions.

(3) Input/output conventions.

Conventions for which modules will perform I/O, how it will be accomplished, and the I/O formats are to be established and followed. The measure is based on which modules do not comply with the conventions.

(4) Error handling conventions.

A consistent method for error handling is required. Conventions established in design are followed into implementation. The measure is based on the number of modules which do not comply with the conventions.

Criteria: Consistency

Metric: CS.2 Data Consistency Measure.

The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Standard data usage representation.

In concert with CS.1 (1), a standard design representation for data usage is to be established and followed. This is a design metric only, identifying the number of modules which violate the standards.

(2) Naming conventions.

Naming conventions for variables and modules are to be established and followed.

(3) Consistent global definitions.

Global data elements are to be defined in the same manner by all modules. The measure is based on the number of modules in which the global data elements are defined in an inconsistent manner for both design and implementation.

(4) Requirements for verifying database consistency/concurrency.

In a system where multiple versions of the same information and data exist at different nodes, requirements should be stated to verify consistency and concurrency of the multiple versions.

(5) Procedures for verifying database consistency/concurrency.

As in (4) above, procedures should be developed for verifying consistency/concurrency of multiple versions.

Criteria: <u>Distributedness</u>

Metric: DI.1 Design Structure Checklist.

The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Design organization identifies all functions and interfaces.
Identification of the complete set of functions and interfaces is essential to the design.

(2) Database organization identifies all data and data flow.
Identification of the complete set of data and flows is essential to the design.

(3) Specific requirements for information distribution within the database.
Early decisions are required on how to distribute information within a network.

(4) Provisions for file/library access from other nodes.
Network nodes will rely on other nodes for some information or for backup data.

(5) Provisions for selecting alternate processing capabilities.
A versatile network design will provide alternate processing sources.

(6) Critical system functions distributed over redundant elements/nodes.
System vulnerabilty is reduced by distributing critical functions across different nodes.

(7) Distribution of control functions ensures network operation/integrity under anomalous conditions.
Again, a good network design will take advantage of the redundant processing capability and distribute network control functions across different nodes.

Criteria: <u>Distributedness</u>

(8) Logical structure and function separated in the design.
Logical entities can be grouped under one function or can be separated among several functions. It is important to distinguish between logical structure and function.

(9) Physical structure and function separated in the design.
Functions can be grouped within one physical structure or can be separated among several physical structures. It is important to distinguish between physical structure and function.

(10) Number of nodes that can be removed and still have each node able to communicate with each remaining node (Kleitman's algorithm).
The node connectivity is the minimum number of nodes whose removal will disconnect the two nodes. If the two nodes have an arc linking them, there is no way to disconnect them by removing nodes, not even by removing all $n - 2$ of the remaining nodes in an n node network. In this case the node connectivity is defined as $n - 1$. If a network can withstand the loss of k nodes, it can also withstand the loss of k links, by Whitney's theorem. An algorithm due to Kleitman (1969) is as follows. Pick any node at random and call it N_1 and every other node in the network is at least $k + 1$.

Now delete N_1 and all its attached links from the network and choose another node, N_2. Verify that this node has at least a node connectivity of k with every other node. Next, remove N_2 and its attached links from the network and choose a third node, N_3. Verify that N_3 has at least a node connectivity of $k - 1$ with each of the remaining nodes. Continue this process until you have verified that some node N_{k+1} is 1-connected to all nodes of the remaining network. At this point the algorithm terminates.

Kleitman, D.: "Methods for Investigating the Connectivity of Large Graphs," IEEE Trans. Circuit Theory, vol. CT-16, pp. 232-233, May 1969.

Criteria: <u>Distributedness</u>

S. Even (1975) has devised another way to check for connectivity k.

Even, S.: Graph Algorithms. Potomac, Md.: Computer Science Press, 1979.

Even, S.: "An Algorithm for Determining Whether the Connectivity of a Graph Is at Least k," SIAM J. Comput., vol. 4, pp. 393-396, Sept. 1975.

Criteria: <u>Document Accessibility</u>

Metric: DA.1 Access No-Control.

(1) Is there no access control to the software document?
This metric provides a measure of the ease of access to software documents.

Metric: DA.2 Well-Structured Documentation.
The metric is the sum of the following applicable elements divided by the number of applicable elements.

(1) Clearly and simply written documents.
When the documents are the more clearly and simply written, the software programs are the easier to understand and are more useful. This is a binary measure.

(2) Neat and carefully drawn software flow charts with adequate information and explanation.
When the documents provide system software flow charts and explain the functions performed, they are more useful. This is a binary measure.

(3) Hierarchical structured table of contents used in documents.
The documents with hierarchical structure will make it easy to skim through until the desired information is found, then read in detail. Then the information in the documents is more accessible. This is a binary measure.

(4) Index system used in documents.
Documents with an index system will make it easier and faster to locate the required information. Then the contents of the documents are more accessible. This is a binary measure.

(5) Separate volumes based on function provided.
See explanation for (3) and (4) above.

Criteria: Document Accessibility

(6) Provide global information about the functional range of the system.
The documents should have global information about the range of the function performed. Then the documents are more useful. This is a binary measure.

(7) Describe the functions performed.
The documents should describe the functions performed in the system. This is a binary measure.

(8) Describe the algorithm used and limitations.
The documents should describe the algorithm and their limitations. Then the user will know if they are applicable or not for the desired application. This is a binary measure.

(9) Describe the relationship between functions.
The documents should describe the relationship between the functions. Then the documents will be more useful. This is a binary measure.

(10) Provide software program listing.
The documents should contain the program source listing. Then the information in the documents is complete. This is a binary measure.

Metric: DA.3 Selective Usability
The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Options available to the user so that selected computation or output feature may be requested.
The software with these options tends to be more reusable. This is a binary measure.

Criteria: <u>Document Accessibility</u>

(2) Modules allow for modifying resource utilization i.e., through use variable dimensioned arrays.
The software allowing resource utilization modification tends to be more reusable. This is a binary measure.

(3) Required new functions can be satisfied by using existing design.
The required functions for the new application can generally be satisfied by adaptation of functions/modules from the existing design. The measure is based on the number of existing functions associated with the required new functions. This is an application-dependent metric.

Criteria: <u>Effectiveness</u>

Metric: EF.1 Performance Requirements.

Performance requirements and limitations specified and allocated to functions/design.

Performance requirements for the system must be broken down and allocated appropriately to the functions and modules during the design. This metric simply identifies if the performance requirements have (1) or have not (0) been allocated during the design.

Metric: EF.2 Iterative Processing Efficiency Measure.

The metric at the module level is the sum of the scores of the following applicable elements divided by the number of elements. At the system level it is an averaged score for all of the modules.

(1) Non-loop dependent computations kept out of loop.

Such practices as evaluating constants in a loop are to be avoided. This measure is based on the number of non-loop dependent statements found in all loops in a module. This is to be measured from a detailed design representation during design and from the code during implementation.

(2) Performance optimizing compiler/assembly language used.

This is a binary measure which identifies if a performance optimizing compiler was used (1); or if assembly language was used to accomplish performance optimization (1); or if neither were used (0).

(3) Compound expressions defined once (implementation only).

Repeated compound expressions are to be avoided from an efficiency standpoint. This metric is based on the number of compound expressions which appear more than once.

(4) Number of overlays.

The use of overlays requires overhead with respect to processing time.

Criteria: Effectiveness

This measure, the reciprocal of the number of overlays, reflects that overhead. It can be applied during design, when the overlay scheme is defined, and during implementation.

(5) Free of bit/byte packing/unpacking in loops.
This is a binary measure indicating the overhead involved in bit/byte packing and unpacking. Placing these activities within loops should be avoided if possible.

(6) Module linkages.
This measure essentially represents the inter-module communication overhead. The measure is based on the amount of execution time spent during module-to-module communication.

(7) Operating system linkages.
This measure represents the module to OS communication overhead. The measure is based on the amount of execution time spent during module to OS communications.

Metric: EF.3 Data Usage Efficiency Measure.
The metric at the module level is the sum of the scores of the following applicable elements divided by the number of applicable elements. The system metric is the averaged value of all of the module metric values.

(1) Data grouped for efficient processing.
The data utilized by any module is to be organized in the data base, buffers, arrays, etc., in a manner which facilitates efficient processing. The data organization during design and implementation is to be examined to provide this binary measure.

Criteria: Effectiveness

(2) Variables initialized when declared.
This measure is based on the number of variables used in a module which are not initialized when declared. Efficiency is lost when variables are initialized during executio^ of a function or repeatedly initialized during iterative processing.

(3) No mix-mode expressions.
Processing overhead is consumed by mix-mode expre sions which are otherwise unnecessary. This measure is based on the number of mix-mode expressions found in a module.

(4) Common choice of units/types.
For similar reasons as expressed in (3) above this convention is to be followed. The measure is the reciprocal of the number of operations performed which have uncommon units or data types.

(5) Data indexed or referenced for efficient processing.
Not only the data organization, (1) above, but the linkage scheme between data items effects the processing efficiency. This is a binary measure of whether the indexing utilized for the data was chosen to facilitate processing.

(6) Static data.
This metric measures the numbers of data items which were referenced but not modified during execution.

(7) Dynamic data.
This metric measures the number of data items which were modified during execution.

Criteria: Effectiveness

Metric: EF.4 Storage Efficiency Measure.

The metric at the module level is the sum of the scores of the following applicable elements divided by the number of applicable elements. The metric at the system level is the averaged value of all of the module metric values.

(1) Storage requirements allocated to design.

The storage requirements for the system are to be allocated to the individual modules during design. This measure is a binary measure of whether that allocation is explicitly made (1) or not (0).

(2) Virtual storage facilities used.

The use of virtual storage or paging techniques enhances the storage efficiency of a system. This is a binary measure of whether these techniques are planned for and used (1) or not (0).

(3) Common data defined only once.

Often, global data or data used commonly are defined more than once. This consumes storage. This measure is based on the number of variables that are defined in a module that have been defined elsewhere.

(4) Program segmentation.

Efficient segmentation schemes minimize the maximum segment length to minimize the storage requirement. This measure is based on the maximum segment length. It is to be applied during design when estimates are available and during implementation.

(5) Dynamic memory management utilized.

This is a binary measure emphasizing the advantages of using dynamic memory management techniques to minimize the amount of storage required during execution. This is planned during design and used during implementation.

Criteria: <u>Effectiveness</u>

(6) Data packing used.
While data packing was discouraged in EF.2 (5) in loops because of the overhead it adds to processing time, in general it is beneficial from a storage efficiency viewpoint. This binary measure applied during implementation recognizes this fact.

(7) Storage optimizing compiler/assembly language used.
This binary measure is similar to EF.2 (2) except from the viewpoint of storage optimization.

(8) Database files/libraries stored at only one node.
Avoiding multiple files/libraries increases system storage optimization.

Criteria: Functional Overlap

Metric: FO.1 Functional Overlap Measure.

This metric refers to the overlap of functional responsibility or computation between the two systems that must interoperate. The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Number of duplicated functions in the system that are to interoperate.
When two systems must be made to interoperate, functions which are duplicated in both systems must be examined to determine any potential conflict. This examination for function conflict will require additional effort to assess the two functions and the impact each may have on the other when the systems interoperate.

(2) Number of duplicate functions to be deleted in one or the other system.
The presence of the same functions being implemented or accomplished in both systems is not necessarily detrimental to interoperability, especially if each function remains independent of the other and there is no need to communicate. However, if one of the systems is assigned unique responsibility for that function, and the corresponding function is to be deleted from the other system, then the amount of work to achieve interoperability is increased.

(3) Number of duplicated function pairs to be synchronized.
If the duplicated functions in each system must be synchronized, then the effort to achieve interoperation will be greater than that in (2) because the problems of synchronization are usually more complex than those of deleting one function. Various timing, format, content, and operational considerations may arise while attempting synchronization of the two systems.

Criteria: Functional Overlap

(4) Number of duplicated function pairs requiring redundancy management logic to combine them.

The most complex resolution of duplicated functions is the use of a redundancy management scheme. This calls not only for intimate communication between the duplicated functions, but also calls for complex and intricate logic to resolve apparent differences, identify malfunctions, and determine and implement a reconfiguration approach.

Criteria: Functional Scope

Metric: FS.1 Function Specificity

The degree to which all modules in the system perform single integral well defined functions. The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Number of functions performed per module.
A module ideally should perform a single integral function. This measure is based on the number of functions performed in a module.

(2) Each module has code comments about functional description.
Comments about functions performed in the module are extremely valuable to the person who wants to reuse this module. The measure is based on the number of modules which do not comply.

Metric: FS.2 Function Commonality

This metric refers to the usefulness, to other applications, of the functions performed by the software. The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Is the function constructed in a manner which facilitates or encourages its use elsewhere either in part or in total?
The software constructed in the above manner tends to be more reusable. This is a binary measure.

(2) Are the input quantities well defined?
When input quantities are well defined, the reuse task is easier. This is a binary measure.

(3) Are the input formats well defined?
See explanation for (2) above.

Criteria: Functional Scope

(4) Are the outputs or database well defined and easy to interpret?
A similar explanation to (2) above is applicable here.

(5) Does the function performance satisfy one of the specified requirements?
This is an application dependent metric.

Metric: FS.3 Function Completeness
The degree to which a system performs a total function in terms of user need. This is an application depe.'dent metric.

(1) Number of function requirements satisfied in the specified requirements.
The metric is the number of user requirements satisfied divided by the total number of user requirements. The value is computed for the system metric.

Criteria: <u>Generality</u>

Metric: GE.1 Module Reference By Other Modules.

(1) Number of modules which are referenced by other modules.

This metric provides a measure of the generality of the modules as they are used in the current system. A module is considered to be more general in nature if it is used (referenced) by more than one module. The number of these common modules divided by the total number of modules provides the measure.

Metric: GE.2 Implementation for Generality Checklist.

This metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Input, processing, output functions are not mixed in a single function.

A module which performs I/O as well as processing is not as general as a module which simply accomplishes the processing. This measure is based on the number of modules that violate this concept at design and implementation.

(2) Application and machine dependent functions are not mixed in a single module.

Any references to machine dependent functions within a module lessens its generality. An example would be referencing the system clock for timing purposes. This measure is based on the number of machine dependent functions in a module.

(3) Processing not data volume limited.

A module which has been designed and coded to accept no more than 100 data item inputs for processing is certainly not as general in nature as a module which will accept any volume of input. This measure is based on the number of modules which are designed or implemented to be data volume limited.

Criteria: <u>Generality</u>

(4) Processing not data value limited.

A previously identified element, AM.2 (2) of Anomaly Management dealt with checking input for reasonableness. This capability is required to prevent providing data to a function for which it is not defined or its degree of precision is not acceptable, etc. This is necessary capability from an error tolerance viewpoint. From a generality viewpoint, the smaller the subset of all possible inputs to which a function can be applied the less general it is. Thus, this measure is based on the number of modules which are data value limited.

Criteria: Independence

Metric: ID.1 Software System Independence Measure

The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Dependence on software system utility programs, system library routines, and other system facilities.

The more utility programs, library routines, and other system facilities that are used within a system, the more dependent the system is on that software system environment. A SORT utility in one operating system is unlikely to be exactly similar to a SORT utility in another. This measure is based on the number of references to system facilities in a module divided by the total number of lines of code in the module.

(2) Common, standard subset of language used

The use of nonstandard constructs of a language that may be available from certain compilers cause conversion problems when the software is moved to a new software system environment. This measure represents that situation. It is based on the number of modules which are coded in a non-standard subset of the language. The standard subset of the language is to be established during design and adhered to during implementation.

Metric: ID.2 Machine Independence Measure

The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Programming language used available on other machines.

This is a binary measure identifying if the programming language used is available (1) on other machines or not (0). This means the same version and dialect of the language.

(2) Free from input/output references.

Input and output references bind a module to the current machine configuration. Thus the fewer modules within a system that contain input and output references, the more localized the problem becomes when conversion is considered. This measure represents that fact and is based on the number of I/O references within a module.

(3) Code is independent of word and character size
Instructions or operations which are dependent on the word or character size of the machine are to be either avoided, or parametric, to facilitate use on another machine. This measure, applied to the source code during implementation, is based on the number of modules which contain violations to the concept of independence of word and character size.

(4) Data representation machine independent
The naming conventions (length) used are to be standard or compatible with other machines. This measure is based on the number of modules which contain variables which do not conform to standard data representations.

Criteria: Modularity

Metric: MO.2 Modular Implementation Measure.

The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Hierarchical structure.

The measure refers to the modular implementation of the top down design structure mentioned in SI.1 (1). The hierarchical structure obtained should exemplify the following rule: interactions between modules are retricted to flow of control between a predecessor module and its immediate successor modules. This measure is based on the number of violations to this rule.

(2) Module size profile.

The standard module size of procedural statements can vary. 100 statements has been mentioned in the literature frequently. This measure is based on the number of procedural statements in a module.

(3) Controlling parameters defined by calling module.

The next four elements further elaborate on the control and interaction between modules referred to by (1) above. The calling module defines the controlling parameters, any input data required, and the output data required. Control must also be returned to the calling module. This measure is based on the number of calling parameters which are control parameters. The next three are based on whether a rule is violated. They can all be measured at design and implementation.

(4) Input data controlled by calling module.

See (3) above.

(5) Output data provided to calling module.

See (3) above.

Criteria: <u>Modularity</u>

(6) Control returned to calling module.
See (3) above.

(7) Modules do not share temporary storage.
This is a binary measure, (1) if modules do not share temporary storage and (0) if they do. It emphasizes the loss of module independence if temporary storage is shared between modules.

(8) Each module represents one function.
Ideally, each module performs only one function.

Metric: MO.3 Modular Design Measure.
The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Processes/functions/modules have loose coupling.
In achieving a highly modular design it is essential to minimize the relationships among modules. The goal is to design modules with low coupling. The scale of coupling from worst to best is: 1) content coupling, 2) common coupling, 3) external coupling, 4) control coupling, 5) stamp coupling, and 6) data coupling.

1) Content coupling - one module makes reference to the contents of another module.
2) Common coupling - modules reference a shared global data structure.
3) External coupling - modules reference the same externally declared symbol.
4) Control coupling - one module passes elements of control as arguments to another module.
5) Stamp coupling - two modules reference the same data structure, which is not global.
6) Data coupling - one module calls another and the modules are not coupled as defined above (in 1 through 5).

Criteria: <u>Modularity</u>

(2) Processes/functions/modules have high cohesion.
In achieving a highly modular design it is essential to maximize the relationships among the elements of each module. The following are relative values for seven types of cohesion:

	COHESION TYPE	VALUE
7)	Functional	1.0
6)	Informational	0.7
5)	Communicational	0.5
4)	Procedural	0.3
3)	Classical	0.1
2)	Logical	0.1
1)	Coincidental	0.0

The following are descriptions of the seven types of cohesion.

1) Coincidental
   . No meaningful relationships among the elements of a module.
   . Difficult to describe the module's function(s).

2) Logical
   . Module performs (at each invocation) one of a class of related functions (e.g., "edit all data").
   . Module performs more than one function.

3) Classical
   . Module performs one of a class of functions that are related in time (Program procedure).
   . Module performs more than one function.

4) Procedural
   . Module performs more than one function, where the functions are related with respect to the procedure of the problem (Problem procedure).

5) Communicational
   . Module has procedural strength; in addition, all of the elements "communicate" with one other (e.g., reference same data or

pass data among themselves).
- All functions use the same data.

6) Informational
- Module performs multiple functions where the functions (entry points in the module) deal with a single data structure.
- Physical packaging together of two or more modules having functional strength.
- All functions use the same data.

7) Functional
- All module elements are related to the performance of a single function.

Reference:

For a more detailed explanation of the terms used to describe cohesion and coupling see "Reliable Software Through Composite Design", Myers, Glenford J.

Criteria: Operability

Metric: OP.1 Operability Checklist.

The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) All steps of operation described (normal and alternative flows).
This binary measure identifies whether the operating characteristics have been described in the requirements specification, and if this description has been transferred into an implementable description of the operation (usually in an operator's manual). The description of the operation should cover the normal sequential steps and all alternative steps.

(2) All error conditions and responses appropriately described to operator.
The requirement for this capability must appear in the requirements specification, must be considered during design, and coded during implementation. Error conditions must be clearly identified by the system. Legal responses for all conditions are to be either documented and/or prompted by the system. This is a binary measure to trace the evolution and implementation of these capabilities.

(3) Provisions for operator to interrupt, obtain operational status, save, modify, and continue processing.
The capabilities provided to the operator must be considered during the requirements phase and then designed and implemented. Examples of operator capabilities include halt/resume and check pointing. This is a binary measure to trace the evolution of these capabilities.

(4) Number of operator actions reasonable (requires execution).
The number of operator errors can be related directly to the number of actions required during a time period. This measure is based on the amount of time spent requiring manual operator actions divided by the total time required for the job.

(5) Job set up and tear down procedures described.
The specific tasks involved in setting up a job and completing it are to

be described. This is usually documented during the implementation phase when the final version of the system is fixed. This is a binary measure of the existence of that description.

(6) Hard copy log of interactions maintained.
This is a capability that must be planned for in design and coded during implementation. It assists in correcting operational errors, improving efficiency of operation, etc. This binary measure identifies whether it is considered in the design and implementation phases (1) or not (0).

(7) Operator messages consistent and responses standard.
This is a binary measure applied during design and implementation to insure that the interactions between the operator and the system are simple and consistent. Operator responses such as YES, NO, GO, STOP, are concise, simple, and can be consistently used throughout a system. Lengthy, differently formatted responses not only provide difficulty to the operator but also require complex error checking routines.

(8) Access violations and responses appropriately described.
Appropriate decriptions and a log of access violations will enable the operator to clearly assess the system status.

(9) Capability for operator to obtain network resource status.
This capability is essential for managing individual nodes resources and for providing services which are dependent on other nodes.

(10) Capability to select different nodes for different types of processing or for different types of information retrieval.
This provision expands the virtual capability and versatility of the node.

Criteria: <u>Reconfigurability</u>

Metric: RE.1 Restructure Checklist.

The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Configuration of communication links is such that failure of one node/link will not disable communication among other nodes.
Alternate communication paths ensure the ability to reconfigure the network in the event of a single point failure.

(2) Specific requirements for maintaining data base integrity under anomalous conditions.
In a network where information is distributed among different nodes, and sometimes duplicated at different nodes, it is essential to maintain the integrity of the total database when conditions are non-normal.

(3) Provisions for maintaining database integrity under anomalous conditions.
A scheme is required for implementing the requirements referenced in (2) during the Preliminary Design phase.

(4) Node can rejoin the network when it has been recovered.
It is desirable to have a node rejoin the network without interrupting basic or critical network functions.

(5) Data replicated at two or more distinct nodes.
Information, especially critical data, should be replicated within the system to insure the ability to reconfigure.

Criteria: Self Descriptiveness

Metric: SD.1 Quantity of Comments.

The metric is the number of comment lines divided by the total number of lines in each module. Blank lines are not counted. The average value is computed for the system level metric.

(1) Number of lines of source code and non-blank comments.

Metric: SD.2 Effectiveness of Comments Measure.

The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Modules have standard formatted prologue comments.
This information is extremely valuable to new personnel who have to work with the software after development, performing maintenanc , testing, changes, etc. The measure at the system level is based on the number of modules which do not comply with a standard format or do not provide complete information.

(2) Comments set off from code in uniform manner.
Blank lines, bordering asterisks, specific card columns are some of the techniques utilized to aid in the identification of comments. The measure is based on the number of modules which do not follow the conventions established for setting off the comments.

(3) All transfers of control and destinations commented.
This form of comment aids in the understanding and ability to follow the logic of the module. The measure is based on the number of modules which do not comply.

(4) All machine dependent code commented.
Comments associated with machine dependent code are important not only to explain what is being done but also serves to identify that portion of the module as machine dependent. The metric is based on the

Criteria: <u>Self Descriptiveness</u>

number of modules which do not have the machine dependent code commented.

(5) All non-standard HOL statements commented.
See explanation for (4) above.

(6) Attributes of all declared variables commented.
The usage, properties, units, etc., of variables are to be explained in comments. The measure is based on the number of modules which do not follow this practice.

(7) Comments do not just repeat operation described in language.
Comments are to describe why, not what. A comment, increment A by 1, for the statement A=A+1 provides no new information. A comment, increment the table look-up index, is more valuable for understanding the logic of the module. The measure is based on the number of modules in which comments do not explain the why's.

Metric: SD.3 Descriptiveness of Language Measure.
The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) High order language used.
An HOL is much more self-descriptive than assembly language. The measure is based on the number of modules which are implemented, in whole or part, in assembly or machine language.

(2) Variable names (mnemonics) descriptive of physical or functional property represented.
While the metric appears very subjective, it is quite easy to identify if variable names have been chosen with self-descriptiveness in mind.

Criteria: Self Descriptiveness

Three variable names such as NAME, POSIT, SALRY are far better and more easily recognized as better than A1, A2, A3. The measure is based on the number of modules which do not utilize descriptive names.

(3) Source code logically blocked and indented.
Techniques such as blocking, paragraphing, indenting for specific constructs are well established and are to be followed uniformly with a system. This measure is based on the number of modules which do not comply with a uniform technique.

(4) One statement per line.
The use of continuation statements and multiple statements per line causes difficulty in reading the code. The measure is the number of continuations plus the number of multiple statement lines divided by the total number of lines for each module and then averaged over all of the modules in the system.

(5) Standard format for organization of modules.
All modules should be similar in structure to ease understanding.

(6) No language keywords used as names.
Names should be unique and not include language keywords.

Criteria: <u>Simplicity</u>

Metric: SI.1 Design Structure Measure.
The metric is the sum of the scores of the applicable elements divided by the number of applicable elements.

(1) Design organized in top down fashion.
A hierarchy chart of system modules is usually available or easy to construct from design documentation. It should reflect the accepted notion of top down design. The system is organized in a hierarchical tree structure, each level of the tree represents lower levels of detail descriptions of the processing.

(2) Module independence.
The processing done within a module is not to be dependent on the source of input or the destination of the output. This rule can be applied to the module description during design and the coded module during implementation. The measure for this element is based on the number of modules which do not comply with this rule.

(3) Module processing not dependent on prior processing.
The processing done within a module is not to be dependent upon knowledge or results of prior processing, e.g., the first time through the module, the nth time through, etc. This rule is applied as above at design and implementation.

(4) Each module description includes input, output, processing, limitations.
Documentation which describes the input, output, processing, and limitations for each module is to be developed during design and available during implementation. The measure for this element is based on the number of modules which do not have this information documented.

Criteria: Simplicity

(5) Each module has single entrance, single exit.
Determination of the number of modules that violate this rule at design and implementation can be made and is the basis for the metric.

(6) Size of data base.
The size of the data base in terms of the number of unique data items contained in the data base relates to the design structure of the software system. A data item is a unique data element for example an individual data entry or data field.

(7) Compartmentalization of data base
The structure of the data base also is represented by its modularization or how it is decomposed. The size determined in (6) above divided by the number of data sets provided this measure. A data set corresponds to the first level of decomposition of a data base, e.g., a set in a CODASYL data base, a record in a file system, a COMMON in FORTRAN, or a Data Block in a COMPOOL system

(8) Programming standard developed.
A standard for programming practices will enhance uniformity in module development.

(9) Module descriptions include identification of module interfaces.
Both internal and external interfaces need to be identified.

Metric: SI.2 Structured Language or Preprocessor.

(1) Structured language or preprocessor used.
The use of a structured language or a preprocessor simplifies the programming task.

Criteria: Simplicity

Metric: SI.3 Data and Control Flow Complexity Measure

(1) Complexity measure.

(a) Number of decision points

(b) Number of branching points

The metric measure is the reciprocal of the number branching and decision points.

Metric: SI.4 Coding Simplicity Measure.

The metric at the system level is an averaged quantity of all the module measures for the system. The module measure is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Module flow top to bottom.

This is a binary measure of the logic flow of a module. If it flows top to bottom, it is given a value of 1, if not a 0.

(2) Negative Boolean or complicated compound Boolean expressions used.

Compound expressions involving two or more Boolean operators and negation can often be avoided. These types of expressions add to the complexity of the module. The measure is based on the number of these complicated expressions per executable statement in the module.

(3) Jumps in and out of loops.

Loops within a module should have one entrance and one exit. This measure is based on the number of loops which comply with this rule divided by the total number of loops.

(4) Loop index modified.

Modification of a loop index not only complicates the logic of a module but causes severe problems while debugging. This measure is based on the number of loop indices which are modified divided by the total number of loops.

Criteria: <u>Simplicity</u>

(5) Module is not self-modifying.

If a module has the capability to modify its processing logic it becomes very difficult to recognize what state it is in when an error occurs. In addition, static analysis of the logic is more difficult. This measure emphasizes the added complexity of self-modifying modules.

(6) Number of statement labels.

This measure is based on the premise that as more statement labels are added to a module the more complex it becomes to understand.

(7) Nesting level.

The greater the nesting level of decisions or loops within a module, the greater the complexity. The measure is the reciprocal of the maximum nesting level.

(8) Number of branches.

The more paths or branches that are present in a module, the greater the complexity. This measure is based on the number of decision statements per executable statements.

(9) Statement simplicity level.

This measure is based on the number of declarative and data manipulation statements per executable statement.

(10) Variable mix in a module.

From a simplicity viewpoint, local variables are far better than global variables. This measure is the ratio of internal (local) variables to total (internal (local) plus external (global)) variables within a module.

(11) Variable density.

The more variables used in a module the greater the complexity of that module. This measure is based on the number of variable uses in a module divided by the maximum possible uses.

Criteria: Simplicity

(12) Single use of variables.
Each variable should have a singular use.

(13) Code written according to programming standard.
Uniform module construction and coding conventions aid in minimizing complexity.

(14) Macros and subroutines used to avoid repeated and redundant code.
Use of macros and subroutines is yet another way of simplifying code.

Criteria: Specificity

Metric: SP.1 Scope of Function Measure.

The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Input density.

The fewer the input parameters, the more likely the module is singular in function.

(2) Output density.

The smaller the ratio of output parameters to output values, the more likely the module is singular in function.

(3) Same function cannot be accomplished by multiple variant forms.

If the same function could be accomplished by multiple different modules, the module would not be singular in function.

Criteria: System Accessibility

Metric: SA.1 Access Control Checklist.

The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) User I/O access controls provided.

Requirements for user access control must be identified during the requirements phase. Provisions for identification and password checking must be designed and implemented to comply with the requirements. This binary measure identifies whether attention has been placed on this area.

(2) Data base access controls provided.

This binary measure identifies whether requirements for data base controls have been specified and designed and the capabilities implemented. Examples of data base access controls are authorization tables and privacy locks.

(3) Memory protection across tasks provided.

Similar to (1) and (2) above, this measure identifies the progression from a requirements statement to implementation of memory protection across tasks. Examples of this type of protection, often times provided to some degree by the operating system, are preventing tasks from invoking other tasks, tasks from accessing system registers, and the use of privileged commands.

(4) Network access controls provided.

Similar to the above, this metric identifies the need for access control for the network to protect both the operation of the network and individual nodes.

Criteria: System Accessibility

Metric: SA.2 Access Audit Checklist.

The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Provisions for recording and reporting access to a node.

A statement of the requirement for this type capability must exist in the requirements specification. It is to be considered in the design specification, and coded during implementation. Examples of the provisions which might be considered would be the recording of terminal and processor linkage, data file accesses, and jobs run by user identification and time.

(2) Provisions for immediate indication of access violations.

In addition to (1) above, access audit capabilities required might include not only recording accesses but immediate identification of unauthorized accesses, whether intentional or not.

Criteria: <u>System Clarity</u>

Metric: SC.1 Interface Complexity.

A software program should reduce the interface complexity and promote the system clarity. The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Number of data items (variable names) used to specify the interface.
The measure is based on the number of data items specified by the interface.

(2) Number of data items passed implicitly across interface via common global data without adequate comments.
The measure is based on the number of data items which are passed implicitly across the interface and without adequate comments explanation.

(3) Number of nesting levels in interface.
The greater the nesting level of the interface, the greater the interface complexity. The measure is the reciprocal of the number of nesting levels.

(4) Number of interface data items with negative qualification.
The procedures returning a "TRUE" upon a failure tend to increase the interface complexity.

(5) Number of data items passed across module interface via module arguments and values or via common global data.
The more data items passed across the interface the more complex the interface. The measure is the reciprocal of the number of data items passed across the interface.

Criteria: System Clarity

(6) Module interfaces established by common control blocks or common data blocks or common overlay region of memory or common I/O devices or global variable names and with adequate comments.
The interface established by common control blocks or common global data is more complex than the interface established by parameter lists. This is a binary measure.

(7) Modules do not modify other modules.
The degree of coupling is higher for modules that modify other modules. The measure is based on the number of modules which do not comply with the rule.

Metric: SC.2 Program Flow Complexity.
Software programs should reduce the program flow complexity and promote the system clarity. The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Number of possible unique excution paths.
The measure is the reciprocal of the number of unique execution paths.

(2) Number of IF statements.
The measure is the reciprocal of the number of IF statements.

(3) Number of function CALLs in each module.
The more function CALLs are present in a module, the greater the complexity. The measure is the reciprocal of the number of function CALLs.

(4) Number of control variables used to direct execution path selection.
The measure is the reciprocal of the number of control variables.

Criteria: System Clarity

(5) Number of DO groups.
The measure is the inverse of the number of DO groups.

(6) Each module has code comments that indicate called-by modules and calling modules.
The measure is based on the number of modules which do not comply.

Metric: SC.3 Application Functional Complexity.
Software program should reduce the application functional complexity and promote the system clarity. The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Separate input/output from computational functions.
The measure is based on the number of modules that violate this rule.

(2) Modules do not share temporary storage locations.
The measure is based on the number of modules that violate this rule.

(3) Separate database-management routines and storage-management routines.
The measure is based on the number of modules that violate this rule.

(4) Common function is not distributed among different modules.
Common functions distributed among several different modules will tend to obscure the program logic in each module. This is a binary measure.

(5) Module is not made to do too many (related but different) functions.
Too many related but different functions in a module will tend to obscure the logic with tests to distinguish among the various functions. This is a binary measure.

Criteria: <u>System Clarity</u>

Metric: SC.4 Communication Complexity.

Software programs should reduce the communication complexity and promote the system clarity. The metric is the sum of the following applicable elements divided by the number of applicable elements.

(1) Number of formal parameters each routine.
The measure is the number of parameters divided by the number of global variables.

(2) Common global variable used each module.
The measure is the reciprocal of the number of common global variables used.

(3) Routine-Global-Routine data binding.
The measure is based on the number of global variables which are modified by one routine and referenced to other routines.

(4) Module connections are established by referring to other modules by their functional names, not internal elements of other modules.
Modules whose connections are established by referring to other modules by their functional names are more loosely coupled than are modules whose connections refer to internal elements of other modules. The measure is based on the number of modules which do not comply.

(5) Communication between modules is by passing data, not by passing control elements.
The measure is based on the number of modules which do not comply.

Metric: SC.5 Structure Clarity.

To remove the program impurities, to improve the structure clarity, and make software easier to understand. The metric is a measure reflecting this improvement and is the sum of the following applicable elements divided by the number of applicable elements.

Criteria: <u>System Clarity</u>

(1) Do not compute the same value more than once.
Whenever a specific combination of terms must be used more than once a new name should be assigned to that combination and that new name should be utilized in the subsequent occurrences of that term. The binary metric measure reflects this readability improvement.

(2) Do not insert a statement which never needs to be executed.
To remove the unwarranted assignment statement and improve the comprehensibility of program. This is a binary measure to reflect this improvement.

(3) Maintain a constant meaning for each variable.
Modules should not use the same variable to represent different types of values in different portions of program to improve the understandability. This is a binary measure to reflect this improvement.

(4) Eliminate unnecessary intermediate variables.
See explanation for (2) above.

Criteria: System Compatibility

Metric: SY.1 Communication Compatibility Checklist.
The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Same I/O transmission rates in both systems.
If the two systems have incompatible transmission rates, extra effort will be required to avoid buffer overruns, data overruns, and lost data. Thus, the effort to interoperate in this case is increased.

(2) Same communication protocol in both systems.
Compatible communication protocols assures the systems can begin to converse. If the protocols are incompatible, then additional work will be required so that the systems can initiate mutual communication.

(3) Same message content in both systems.
If the content of the messages are not the same, that is, the same units, the same variable, the same reference points, and the same reference structure, then the message will have a meaning to the receiver different from that intended by the sender.

(4) Same message structure and sequence in both systems.
Even though the protocols may be compatible, and the data of mutual format and type, interoperation may be impossible if the message structure and message sequences are not compatible.

Metric: SY.2 Data Compatibility Checklist.
The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Is data in both systems in the same format (ASCII, EBCDIC,...)
The format of the data transmitted between the systems should be identical, otherwise, additional effort must be spent converting the

Criteria: System Compatibility

format in one system; or a hardware or software reformatter must be designed and implemented.

(2) Same data base structure in both systems.
If the data base structures are compatible, then consistent accessing and indexing interpretations are possible, lessening the chance of incompatibilities which would increase the effort to achieve interoperation.

(3) Same data base access techniques in both systems.
This metric component is related to (2), but it is unique in that it assures that the accessing variables will be as similar as possible between the systems, reducing the conversion necessary between systems.

Metric: SY.3 Hardware Compatibility Checklist.
The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Same word length in both systems.
If both systems use the same standard word length, then problems of differing accuracy and conversion are removed.

(2) Same interrupt structure in both systems.
If both systems use computers with the same interrupt structure, it is likely that they will be mutually compatible in their interfaces with the real world of sensors, etc.

(3) Same instruction set in both systems.
If both systems use computers with identical instruction sets, then they truly "talk the same language." This compatibility should contribute to reduced effort to achieve interoperation between the two systems.

Criteria: System Compatibility

Metric: SY.4 Software Compatibility Checklist.
The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Same source language in both systems.
If the source language used in the two systems is the same, then many compatibilities are already provided; if not, the effort to interoperate will increase due to resolution of language feature discrepancies.

(2) Same operating system in both systems.
Identical operating systems will provide assurance of consistent features and methods of operation. Thus, the effort required to interoperate should be reduced.

(3) Same support software in both systems.
If identical support software is used for the systems that must interoperate, it is likely that both may be constructed in much the same way. The communication necessary to service both systems will be simplified. Finally, duplicate support software centers may provide greater reliability, or, alternatively, the possibility for cost reductions.

Metric: SY.5 Documentation for Other Systems.

(1) Is the other system documentation available in a form that is up-to-date, complete, and clearly organized and written?
Many questions about the other system will arise in achieving interoperability, and the most efficient and practical way of answering them is the availability of documentation on the other system. For the documentation to be useful, however, it must meet certain requirements. It must reflect the other system as it currently exists, or as it will exist at the time of interoperation; so the documentation must be up-to-date. The documentation must also be complete, at least to the extent necessary to answer all questions relating to interoperability. But, even

the most complete and up-to-date documents will be relatively useless if they are not clearly organized and clearly written. The reader must be able to find his way efficiently to the answer he needs, and when found, the answer must be stated clearly. Otherwise, the time lost to locate and understand the information will be excessive and it is likely the reader will make an assumption for his purposes. Once again, the result is likely to be additional interoperability problems.

Criteria: Traceability

Metric: TR.1 Cross Reference.

(1) Cross reference relating functions/modules to requirements.

During design, the identification of which itemized requirements are satisfied in the design of a module are documented. A traceability matrix is an example of how this can be done. During implementation, which itemized requirements are being satisfied by the module implementation are to be identified. Some form of automated notation, prologue comments or imbedded comments, is used to provide this cross reference. The binary metric is the identification of a tracing of requirements into design and into code.

Criteria: Training

Metric: TN.1 Training Checklist.

The metric is the sum of the scores of the following applicable elements divided by the number of applicable elements.

(1) Lesson plans/training material developed for operators, end users, maintainers.

This is a binary measure of whether this type documentation is provided during the implementation phase.

(2) Realistic simulated exercises provided.

This is a binary measure of whether exercises, which represent the operational environment, are developed during the implementation phase for use in training.

(3) Sufficient 'help' and diagnostic information available on-line.

This is a binary measure of whether the capability to aid the operator in familiarization with the system has been designed and built into the system. Provision of a list of legal commands or a list of the sequential steps involved in a process are examples.

(4) Selectable levels of aid and guidance for users of different degrees of expertise.

This is a binary measure of multi-level capability for user familiarization.

Criteria: Virtuality

Metric: VR.1 System/Data Independence Checklist.
This metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Specific requirements for virtual storage structure.
Requiring a virtual storage structure is the key to providing the user with a virtual system.

(2) Provisions for virtual storage structure (user can obtain data without knowing identity/location of storage device).
During Preliminary Design, a scheme is required to implement the requirements referenced in (1). The scheme may be elaborate if data is widely distributed within the network.

(3) Users can manipulate data as if it were not replicated elsewhere in the system.
This measure refers to potential configuration management problems in a network where the same data is replicated at different nodes.

(4) Each user can utilize the system as though it were dedicated to that user.
Presenting each user with a system which is virtually dedicated to that user maximizes the capabilities available to the user.

(5) User is presented with a complete logical system without regard to physical topology.
Lifting the requirement for the user to know the physical topology of the system simplifies the user's task with respect to the system.

Criteria: <u>Visibility</u>

Metric: VS.1 Module Testing Measure.

The system level metric is an average of all module measures. The module measure is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Path coverage.

Plans for testing the various paths within a module should be made during design and test cases actually developed during implementation. This measure identifies the number of paths planned to be tested divided by the total number of paths.

(2) Input parameters boundary tested.

The other aspect of module testing involves testing the input ranges to the module. This is done by exercising the module at the various boundary values of the input parameters. Plans to do this must be specified during design and coded during implementation. The measure is the number of parameters to be boundary tested divided by the total number of parameters.

Metric: VS.2 Integration Testing Measure.

The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Module interfaces tested.

One aspect of integration testing is the testing of all module-to-module interfaces. Plans to accomplish this testing are prepared during design and the tests are developed during implementation. The measure is based on the number of interfaces to be tested divided by the total number of interfaces.

Criteria: Visibility

(2) Performance requirements (timing and storage) coverage.
The second aspect of integration testing involves checking for compliance at the module and subsystem level with the performance requirements. This testing is planned during design and the tests are developed during implementation. The measure is the number of performance requirements to be tested divided by the total number of performance requirements.

Metric: VS.3 System Testing Measure.
The metric is the sum of the scores given to the following elements divided by the number of applicable elements.

(1) Module coverage (for all test scenarios).
One aspect of system testing which can be measured as early as the design phase is the equivalent to path coverage at the module level. For all system test scenarios planned, the percent of all of the modules to be exercised is important.

(2) Identification of test inputs and outputs in summary form.
The results of tests and the manner in which these results are displayed are very important to the effectiveness of testing. This is especially true during system testing because of the potentially large volume of input and output data. This binary measure simply identifies if the capability exists to display test inputs and outputs in a summary fashion. The measure can be applied to the plans and specifications in the design phase and the development of this capability during implementation.